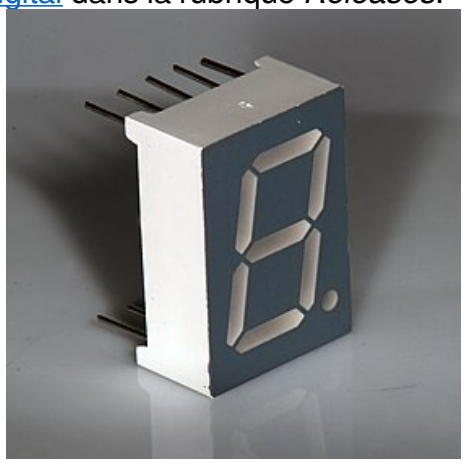
David DEFOUR
Kenelm LOUETSI

/TP N°2
2: Afficheur 7-segments

Outil
Nous utiliserons le logiciel libre ***Digital.*** Il est téléchargeable à : https://github.com/hneemann/Digital dans la rubrique *Releases*. 

*(les questions sont à compléter sur la «**Fiche réponse**» correspondante)*

On cherche à afficher des nombres binaires en un format plus lisible pour l'humain, le décimal. Pour cela nous allons utiliser un afficheur 7-segments.

Exercice 1 : Conversion binaire → BCD

Nous allons tout d'abord chercher à convertir le binaire vers un format qui se rapproche plus du décimal, le *BCD*.

1.1

Pour rappel, le format BCD ou, « **B**inary-**C**oded **D**ecimal », consiste à remplacer chaque chiffre constituant un nombre par son équivalent en binaire.

Par exemple : 135_{10} (décimal) == $1000\ 0111_2$ (binaire) converti en BCD donne :

1	3	5
0001	0011	0101

Combien faut-il de bits pour encoder les dix chiffres arabes en BCD ?

1.2

Donnez la table de conversion décimal → BCD pour les dix chiffres arabes.

1.3

La représentation binaire d'un nombre entier non-signé *d* se fait sur *n* bits, où $n >\approx \log2(d)$.
Inversement, *n* bits permettent de représenter des nombres entiers non-signés allant de 0 à 2^n

Combien faut-il de bits en BCD pour encoder un nombre entier non-signé représentable sur quatre bits en binaire?

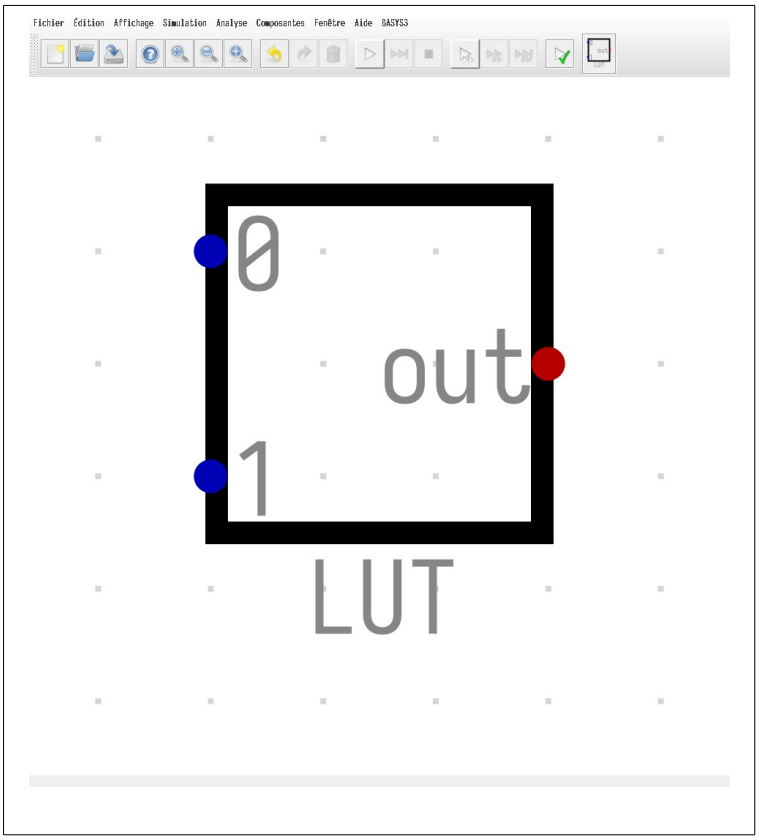
Enseignants : David DEFOUR
Kenelm LOUETSI

1.4

Commençons par implémenter un convertisseur binaire → BCD.

La table de conversion binaire → BCD étant connue, nous allons l'utiliser telle quelle.
Pour cela nous allons faire usage d'un composant permettant d'implémenter des tables, une « *LUT* » (***L**ook-**U**p **T**able*).

Celui-ci se trouve dans : *Composants > Logique > Table de recherche*



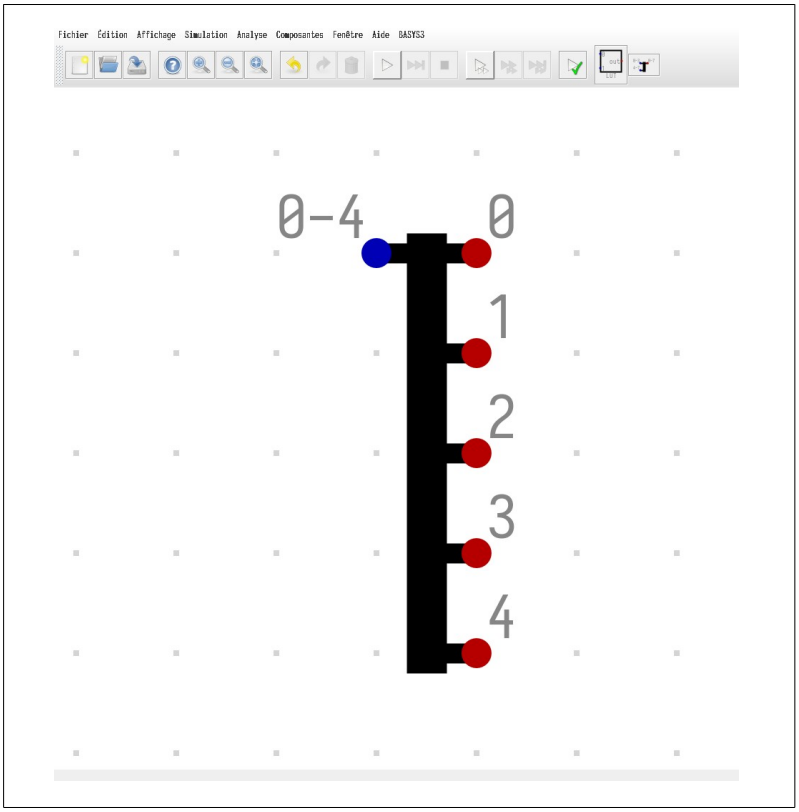
(ce composant ne possède qu'une seule sortie dont la largeur en nombre de bits est configurable)

De quelle largeur (en bits) doit être la sortie de la LUT afin de pouvoir convertir un nombre binaire de 4-bits en BCD ?

1.5

Il est parfois nécessaire d'accéder à un bit précis parmi plusieurs.
Pour cela, est possible de séparer un groupe de bits à l'aide d'un « *Séparateur/Fusionneur* ».

On le trouve dans : *Composants > Cablâge > Séparateur/Fusionneur*



(la syntaxe de configuration de ce composant se trouve dans la rubrique «Aide» du composant)

Notre convertisseur aura pour entrée un nombre binaire de 4-bits ('B3', 'B2', 'B1', 'B0') en « little-endian » et en sortie assez de bits pour pouvoir prendre en compte *toutes* les valeur possible de l'entrée.

Ouvrez le fichier *TP2 > circuits > conv_bin_2_bcd_4bits.dig* dans Digital et dessinez le circuit du convertisseur binaire → BCD.

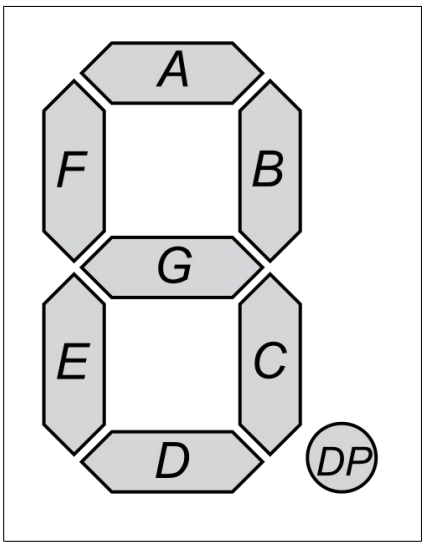
Une fois le circuit validé, enregistrez et fermez le fichier conv_bin_2_bcd_4bits.dig.

Exercice 2 : Conversion BCD → 7-segment

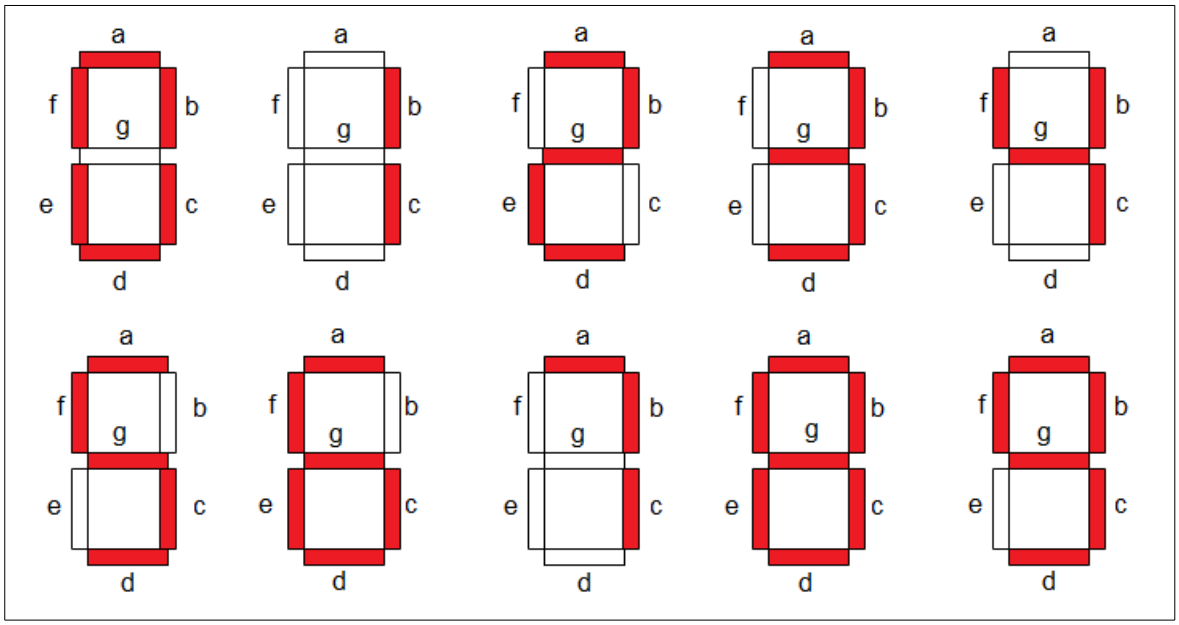
Après avoir fait la conversion binaire → BCD, on peut à présent directement viser l'afficheur 7-segments.

2.1

Nous allons créer un convertisseur BCD → 7-segments.
Un afficheur 7-segments se compose de sept segments ('A', 'B', 'C', 'D', 'E', 'F', 'G', plus un point 'DP') adressables individuellement.



Ces segments permettent d'afficher jusqu'à seize caractères différents, dont les chiffres arabes.



Notre convertisseur BCD → 7-segments va prendre une entrée un *chiffre* encodé en BCD ('S3', 'S2', 'S1', 'S0') et aura pour sortie les sept segments d'un afficheur ('A', 'B', 'C', 'D', 'E', 'F', 'G').

Donnez la table de conversion BCD → 7-segments.

2.2

Tout comme pour le convertisseur binaire → BCD, la conversion BCD → 7-segments utilise une table de conversion fixe.
On peut implémenter celle-ci avec une « *LUT* ».

Ouvrez le fichier *TP2 > circuits > conv_bcd_2_7seg_4bits.dig* dans Digital et dessinez le circuit du convertisseur BCD → 7-segments.

Une fois le circuit validé, enregistrez et fermez le fichier conv_bcd_2_7seg_4bits.dig.

Exercice 3 : Afficheur 7-segments 4-bits

3.1

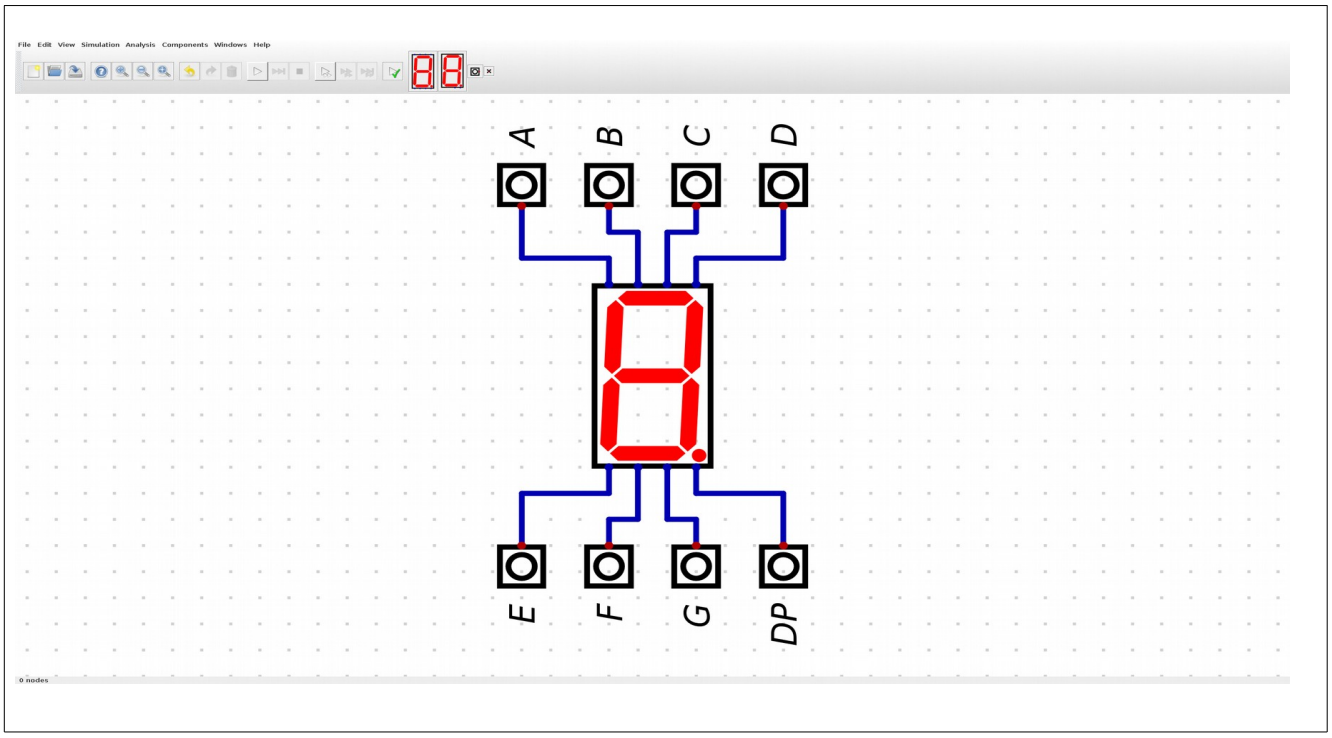
À présent qu'ont été réalisés les convertisseurs binaire → BCD et BCD → 7-segments,
il est possible d'assembler ceux-ci pour afficher la valeur d'un nombre binaire sur un 7-segments.

Ouvrez le fichier *TP2 > circuits > conv_bin_2_7seg_4bits.dig* dans Digital et à l'aide des convertisseurs précédemment crées, dessinez un circuit permettant de convertir un nombre binaire de 4-bits entier non-signé en BCD et de convertir un chiffre BCD en son code 7-segments.

Une fois le circuit validé, enregistrez et fermez le fichier conv_bin_2_7seg_4bits.dig.

3.2

L'afficheur 7-segments se trouve dans : *Composants → E/S → Affichage → Affichage 7 segments*



(puisqu'un ne convertit que des nombres entiers, le segment «DP» ne sera pas utilisé et peut être ignoré)

Ouvrez le fichier *TP2 > circuits > afficheur_bin_2_7seg_4bits.dig* dans Digital et à l'aide des convertisseurs précédemment crées, dessinez un circuit permettant d'afficher la valeur un nombre binaire entier non-signé de 4-bits sur deux afficheurs 7-segments.

Une fois le circuit validé, enregistrez et fermez le fichier afficheur_bin_2_7seg_4bits.dig.

Exercice 4 : Afficheur 7-segments 8-bits

On veut à présent pouvoir afficher un nombre binaire entier non-signé de 8-bits sur des 7-segments.

A) Convertisseur binaire 8-bits → BCD

Il faut pouvoir convertir un nombre binaire entier non-signé de 8-bits en BCD.

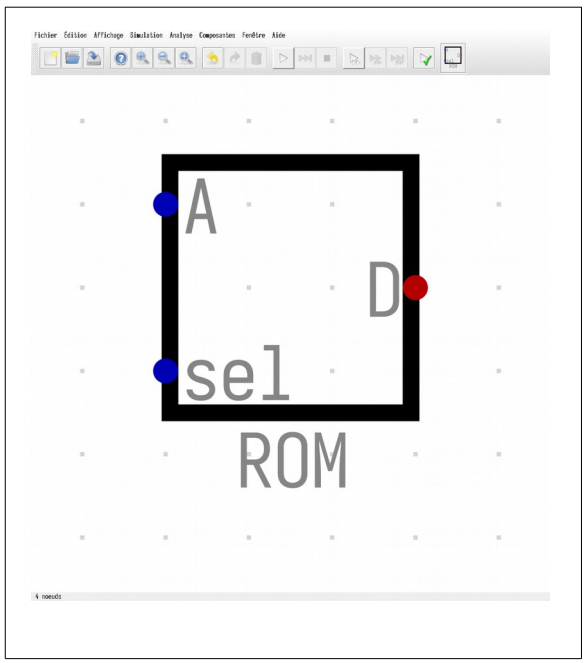
4.A.1
Combien d'afficheurs 7-segments faut-il pour cela ?

4.A.2
On peut enregistrer la conversion de tous les nombres binaires entiers non-signés représentables sur 8-bits dans un tableau.

Quelle est la longueur de la table de conversion 8-bits → BCD ?

4.A.3
Au vu de la longueur de la table de conversion 8-bits→ BCD, on ne va pas stocker celle-ci dans une « *LUT* », mais plutôt dans une mémoire.
Pour effectuer une conversion à partir d'une table déjà remplie, on a seulement besoin de faire des lectures.
Ainsi, nous allons utiliser une mémoire en « lecture seule », une « *ROM* » (**R**ead-**O**nly **M**emory).

Celle-ci se trouve dans : *Composants > Mémoire > ROM*



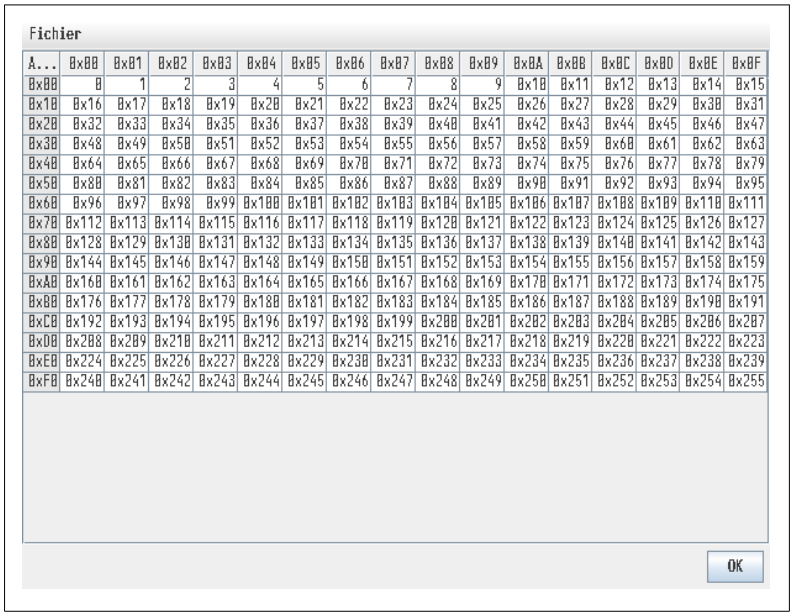
(l'entrée «sel» corresponds au signal d'activation de la sortie «D» de la ROM)

Pour rappel, une mémoire peut être assimilée à un tableau d'éléments.
Éléments qui sont tous indexés par une « adresse » unique.

Pour la conversion 8-bits → BCD, quelle partie de la conversion corresponds le plus naturellement à une « adresse » et laquelle corresponds le plus a un élément associé à une adresse ?

4.A.4

Afin d'éviter d'avoir à écrire toute la table de conversion à la main, celle-ci vous est fournie via le fichier :*TP2 > circuits > bin_2_bcd__8bits_rom.hex.*



(pour charger la table dans le composant ROM:*menu de configuration du composant > Éditer > Fichier > Charger > bin_2_bcd__8bits_rom.hex)*

Ouvrez le fichier *TP2 > circuits > conv_bin_2_bcd_8bits.dig* et dessinez un circuit de convertisseur binaire 8-bits → BCD en utilisant une ROM.

Une fois le circuit validé, enregistrez et fermez le fichier conv_bin_2_bcd_8bits.dig.

B) Afficheur 7-segments 8-bits

Après avoir crée le convertisseur 8-bits → BCD, il nous reste à convertir le BCD obtenu en 7-segments.

4.B.1

Y-a-t-il des modifications à apporter à apporter au convertisseur BCD → 7-segments précédemment réalisé ? Si oui, lesquelles ?

4.B.2

Ouvrez le fichier *TP2 > circuits > conv_bin_2_7seg_8bits.dig* dans Digital et à l'aide des convertisseurs précédemment crées, dessinez un circuit permettant de convertir un nombre binaire entier non-signé en BCD et de convertir un chiffre BCD en son code 7-segments.

Une fois le circuit validé, enregistrez et fermez le fichier conv_bin_2_7seg_4bits.dig.

4.B.3

Ouvrez le fichier *TP2 > circuits > afficheur_bin_2_7seg_8bits.dig* dans Digital et à l'aide des convertisseurs précédemment crées, dessinez un circuit permettant d'afficher la valeur un nombre binaire entier non-signé de 8-bits sur trois afficheurs 7-segments.

Une fois le circuit validé, enregistrez et fermez le fichier afficheur_bin_2_7seg_8bits.dig.