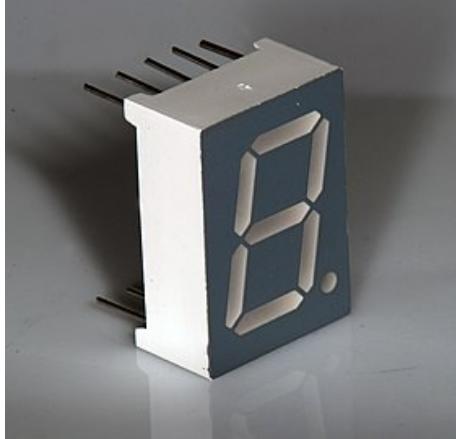


## /TP N°2

### 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, « **Binary-Coded Decimal** », 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  $n \geq \log_2(d)$ .  
Inversement,  $n$  bits permettent de représenter des nombres entiers non-signés allant de 0 à  $2^n$ .

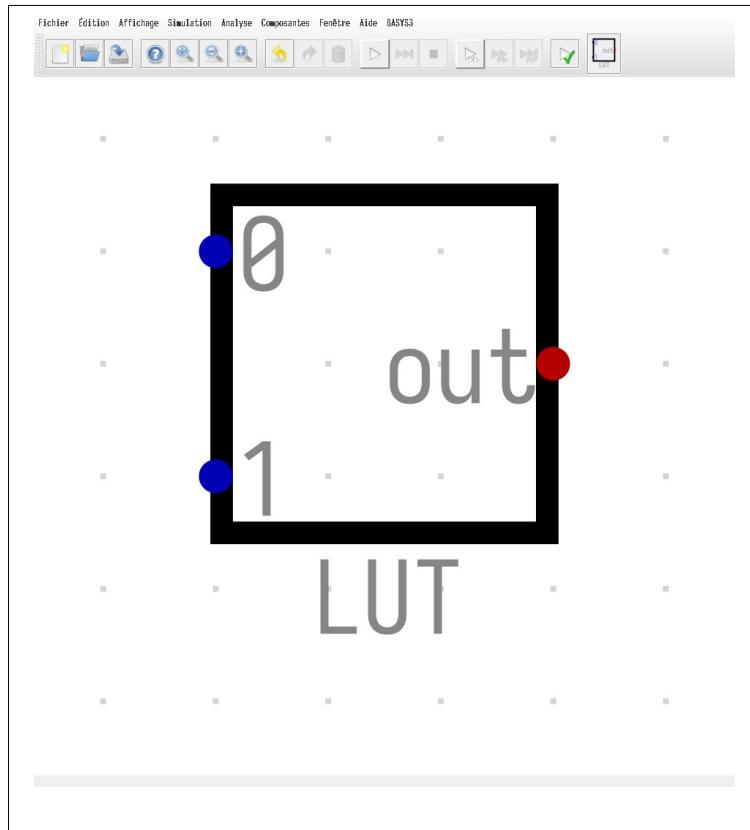
Combien faut-il de bits en BCD pour encoder un nombre entier non-signé représentable sur quatre bits en binaire?

## 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* » (*Look-Up Table*).

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)

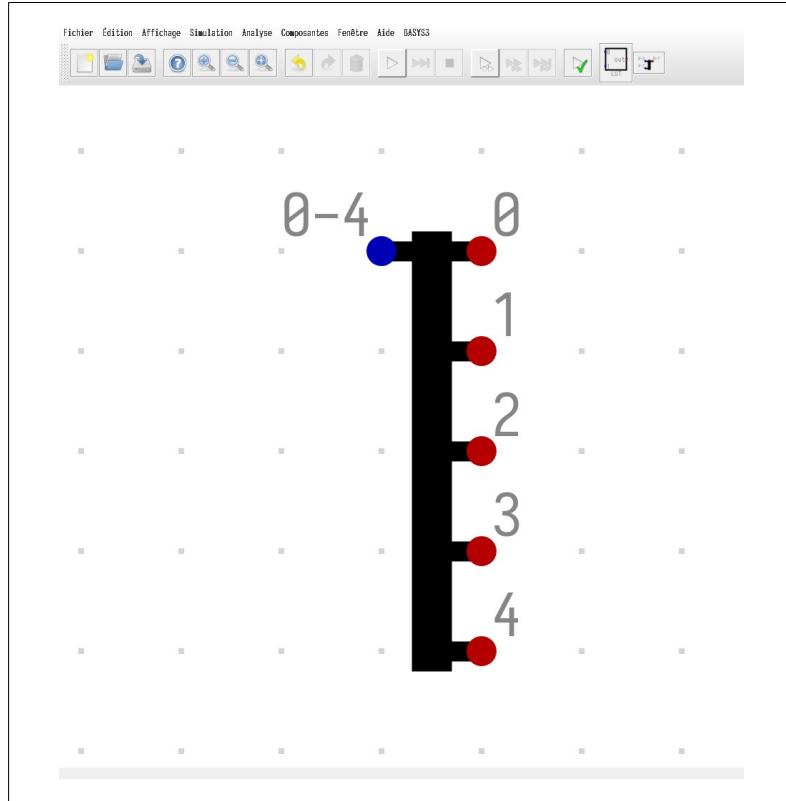
De quelle largeur (en bits) doit être la sortie de la LUT afin de pouvoir convertir un nombre binaire de 4-bits en BCD ?

## 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 > Cablage > 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*.

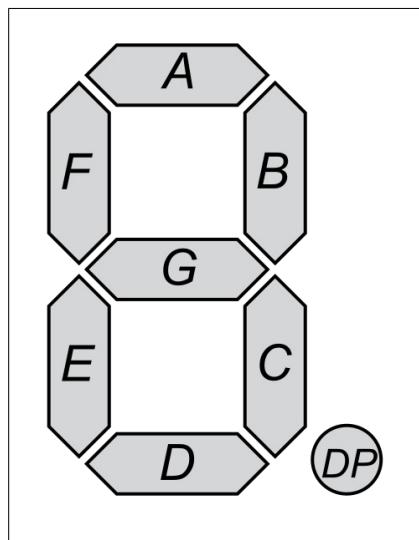
## Exercice 2 : Conversion BCD → 7-segments

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

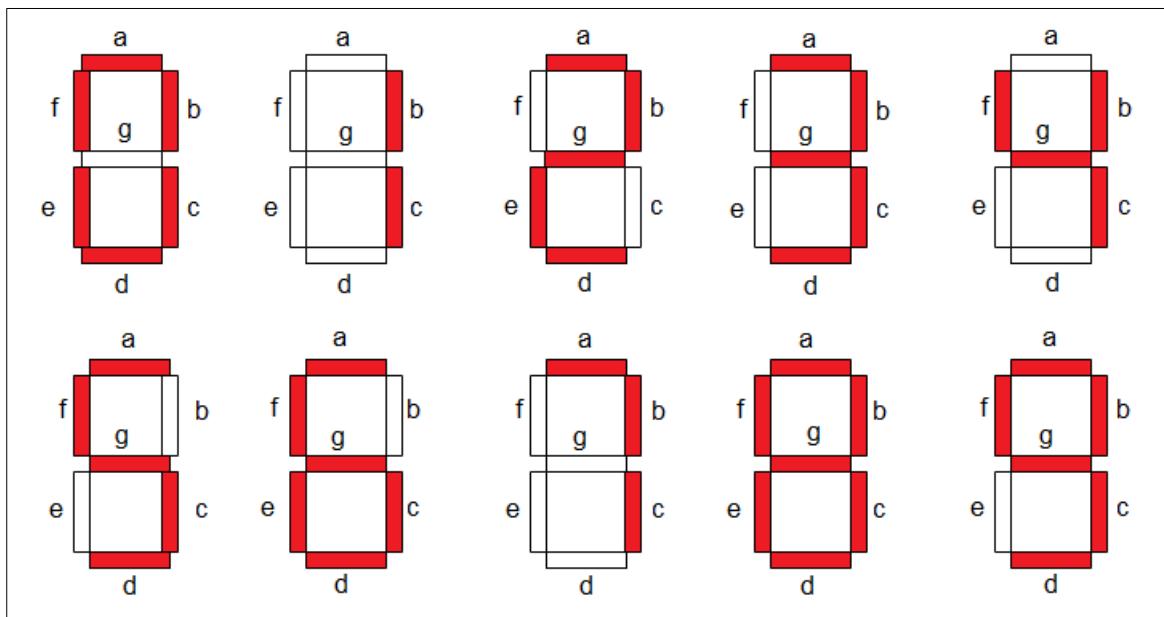
### 2.1

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

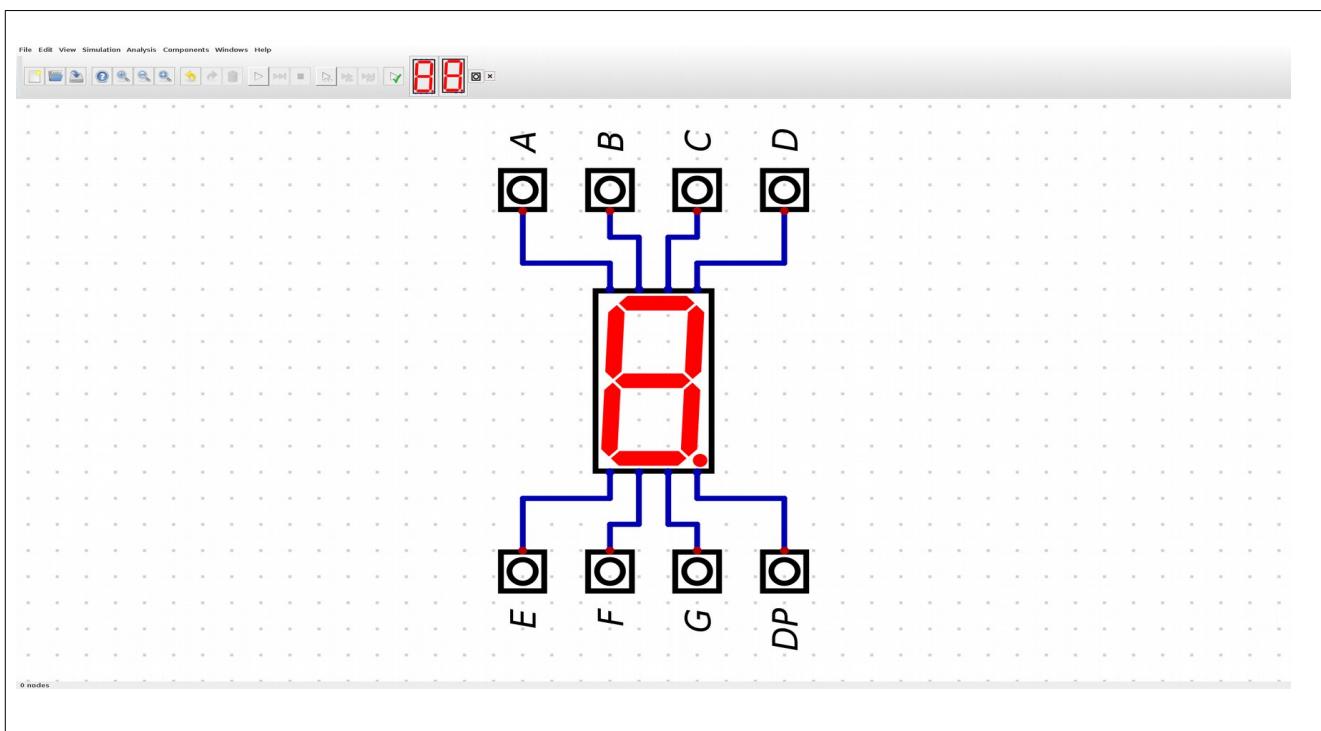
À présent qu'ont été réalisés les convertisseurs binaire → BCD et BCD → 7-segments, il est possible d'assembler ceux-ci pour afficher la valeur d'un nombre binaire sur un 7-segments.

Ouvrez le fichier *TP2 > circuits > conv\_bin\_2\_7seg\_4bits.dig* dans Digital et à l'aide des convertisseurs précédemment créés, 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*



(puisque l'on 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 créés, 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 ?

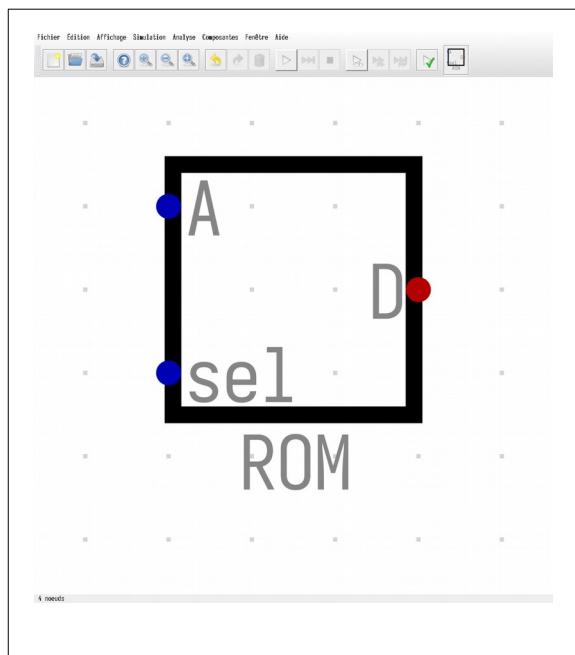
#### 4.A.3

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 à 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.

Fichier																
A...	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
Bx80	0	1	2	3	4	5	6	7	8	9	0x10	0x11	0x12	0x13	0x14	0x15
Bx81	0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31
Bx82	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47
Bx83	0x48	0x49	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57	0x58	0x59	0x60	0x61	0x62	0x63
Bx84	0x64	0x65	0x66	0x67	0x68	0x69	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78	0x79
Bx85	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x90	0x91	0x92	0x93	0x94	0x95
Bx86	0x96	0x97	0x98	0x99	0x100	0x101	0x102	0x103	0x104	0x105	0x106	0x107	0x108	0x109	0x110	0x111
Bx87	0x112	0x113	0x114	0x115	0x116	0x117	0x118	0x119	0x120	0x121	0x122	0x123	0x124	0x125	0x126	0x127
Bx88	0x128	0x129	0x130	0x131	0x132	0x133	0x134	0x135	0x136	0x137	0x138	0x139	0x140	0x141	0x142	0x143
Bx89	0x144	0x145	0x146	0x147	0x148	0x149	0x150	0x151	0x152	0x153	0x154	0x155	0x156	0x157	0x158	0x159
Bx8A	0x168	0x169	0x162	0x163	0x164	0x165	0x166	0x167	0x168	0x169	0x170	0x171	0x172	0x173	0x174	0x175
Bx8B	0x176	0x177	0x178	0x179	0x180	0x181	0x182	0x183	0x184	0x185	0x186	0x187	0x188	0x189	0x190	0x191
Bx8C	0x192	0x193	0x194	0x195	0x196	0x197	0x198	0x199	0x200	0x201	0x202	0x203	0x204	0x205	0x206	0x207
Bx8D	0x208	0x209	0x210	0x211	0x212	0x213	0x214	0x215	0x216	0x217	0x218	0x219	0x220	0x221	0x222	0x223
Bx8E	0x224	0x225	0x226	0x227	0x228	0x229	0x230	0x231	0x232	0x233	0x234	0x235	0x236	0x237	0x238	0x239
Bx8F	0x240	0x241	0x242	0x243	0x244	0x245	0x246	0x247	0x248	0x249	0x250	0x251	0x252	0x253	0x254	0x255

OK

(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 créés, 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 créés, 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*.