INF1500 - Hiver 2011
Laboratoire #3

Affichage alterné d'un nombre en signé et non signé à 4 bits

I- Objectifs

Les objectifs de ce laboratoire sont de décrire et de simuler un circuit de logique combinatoire pour une application numérique. Cet exercice permet de mieux comprendre la représentation de nombres binaires en format signé, non signé et hexadécimal. Dans ce laboratoire, on introduit également l'utilisation d'un module décrit en VHDL (fichier .vhd) et son intégration dans le design.

Ce laboratoire vaut 7 % de la note finale du cours.

II- Travail préparatoire, à réaliser avant le laboratoire (Individuel, 1%)

II.1 Principe de l'affichage à 7 segments

On peut représenter sur 4 bits les nombres hexadécimaux non signés (NS) allant de 0 à F. Selon cet encodage le "bit 0" a un poids de 1 (2^0), le "bit 1" a un poids de 2 (2^1), le "bit 2" a un poids de 4 (2^2), et le "bit 3" a un poids de 8 (2^3).

Par exemple : $(1011)_{2,NS} = 1*8 + 0*4 + 1*2 + 1*1 = (11)_{10} = (B)_{16}$

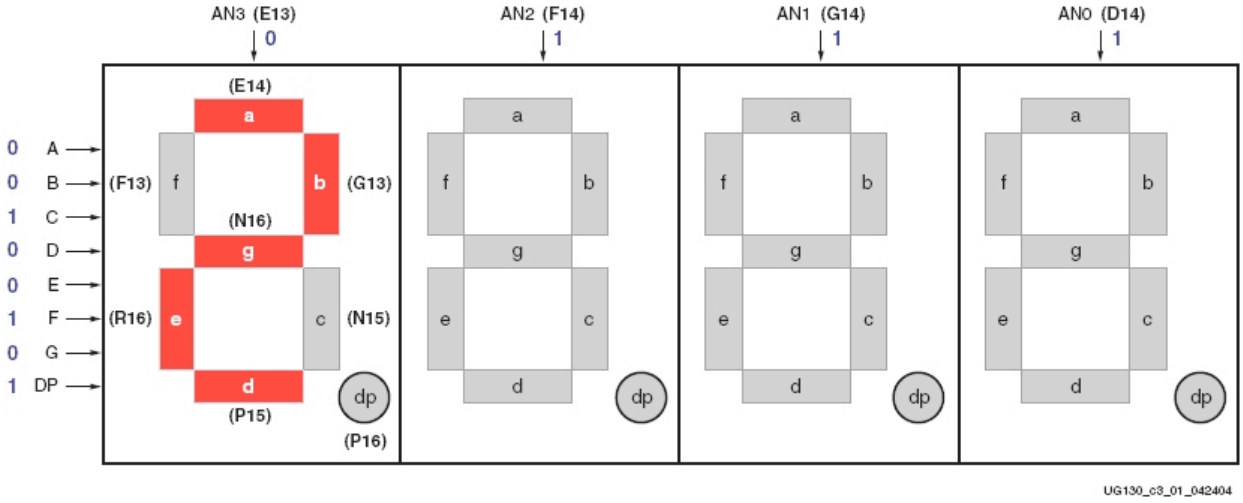
On peut également représenter sur 4 bits les nombres hexadécimaux signés (S) allant de -8 à +7. Dans ce cas le "bit 0" a un poids de 1 (2^0), le "bit 1" a un poids de 2 (2^1), le "bit 2" a un poids de 4 (2^2), et le "bit 3" a un poids de -8 (-2^3). Cette représentation appelée "Complément à deux" (C2) facilite les opérations d'addition et de soustraction.

Par exemple : $(1011)_{2,S} = 1*(-8) + 0*4 + 1*2 + 1*1 = (-5)_{10} = (-5)_{16}$

Le tableau suivant présente la représentation binaire sur 4 bits et son équivalent en format hexadécimal (signé/ non signé) :

Représentation binaire sur 4 bits ($b_3b_2b_1b_0$)	Nombre hexadécimal signé (C2)	Nombre hexadécimal non Signé
0000	+0	0
0001	+1	1
0010	+2	2
0011	+3	3
0100	+4	4
0101	+5	5
0110	+6	6
0111	+7	7
1000	-8	8
1001	-7	9
1010	-6	A
1011	-5	B
1100	-4	C
1101	-3	D
1110	-2	E
1111	-1	F

Pour afficher un chiffre, on peut se servir d'un des quatre afficheurs à 7 segments disponibles sur la planchette de développement. La figure suivante, tirée du manuel de l'utilisateur de la planchette, montre les quatre afficheurs, et spécifiquement comment faire pour afficher le chiffre « 2 » sur l'afficheur de gauche.



On constate que les 4 afficheurs sont composés de 7 segments et d'un point (pour un total, de 8 segments contrôlables). Pour les besoins de ce laboratoire, le point permettra de déterminer si le nombre est négatif (point allumé) ou positif.

Chacun des segments est connecté à une patte du FPGA. Pour allumer un segment particulier, on doit forcer un « 0 » sur la patte correspondante de ce segment, et également forcer un « 0 » sur la patte correspondante de l'anode de l'afficheur en question. L'affichage fonctionne donc en **logique négative**, puisqu'une tension de 0 V fait allumer le segment.

Par exemple, pour afficher le chiffre « 2 » sur l'afficheur de gauche, on doit forcer un « 0 » aux segments a, b, d, e, et g, ainsi que sur l'anode AN_3. Les segments c, f, et le point "dp" doivent quant à eux recevoir un « 1 ».

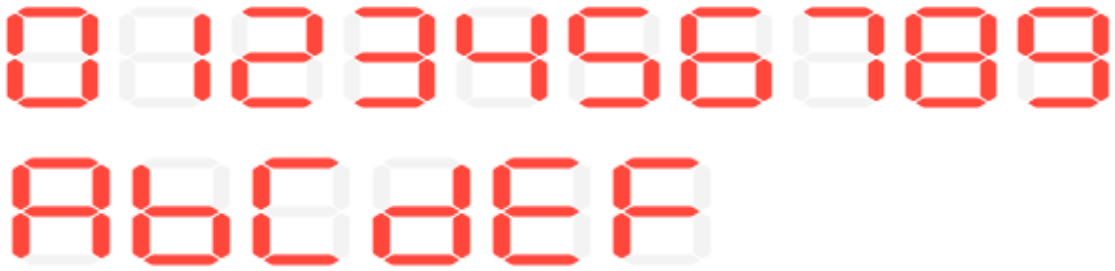
II.2 Travail à réaliser avant le laboratoire

Dans ce laboratoire, vous devez concevoir un circuit de décodage qui accepte 4 bits en entrée (un bus **bit(3:0)**), correspondant au nombre à afficher (**hexadécimal non signé**), et qui a un bus de 8 bits (**ssd(7:0)**) en sortie: un bit pour chacun des 7 segments et un bit pour le point.

Afin de concevoir ce circuit, vous devez remplir la table de vérité à la page suivante, puis, à l'aide de tableaux de Karnaugh, obtenir les équations logiques réduites pour chacune des sorties.

N.B.: Inspecter l'ensemble des équations et des tableaux de Karnaugh dans le but d'identifier des termes communs afin de minimiser la complexité de votre circuit.

Voici les 16 chiffres représentés avec l'affichage 7 segments :



(*source : http://fr.wikipedia.org/wiki/Afficheur_7_segments*)

Important!

Chaque membre de l'équipe doit **remettre** la table de vérité complétée(tableau de la page suivante), les tables de Karnaugh et les équations logiques réduites (les versions manuscrites sont acceptées) au chargé de laboratoire au début de la période. Ce travail préliminaire vaut ***1%*** de la note finale du cours. Prévoyez une **copie supplémentaire** pour l'utiliser dans le laboratoire.

II.2.1 Table de vérité

Nombre	**bit3**	**bit2**	**bit1**	**bit0**	segment 'a' **ssd(0)**	segment 'b' **ssd(1)**	segment 'c' **ssd(2)**	segment 'd' **ssd(3)**	segment 'e' **ssd(4)**	segment 'f' **ssd(5)**	segment 'g' **ssd(6)**	le point 'dp' **ssd(7)**
0	0	0	0	0								
1	0	0	0	1								
2	0	0	1	0								
3	0	0	1	1								
4	0	1	0	0								
5	0	1	0	1								
6	0	1	1	0								
7	0	1	1	1								
8	1	0	0	0								
9	1	0	0	1								
A	1	0	1	0								
b	1	0	1	1								
C	1	1	0	0								
d	1	1	0	1								
E	1	1	1	0								
F	1	1	1	1								

II.2.2 Travail préparatoire non évalué: Circuit d'affichage alterné (signé et non signé)

Vous devez concevoir vous-même un circuit permettant de faire un affichage alterné entre signé (C2) et non signé. *Alors, pensez-y dès maintenant!*

Ce circuit doit avoir en entrée un bus de 4 bits (**bit(3:0)**) représentant le nombre à afficher et un signal de sélection (**R**) permettant de choisir l'un des deux modes de représentation. La sortie de ce circuit reste toujours un bus de 8 bits (**ssd(7:0)**) où chaque bit correspond à un segment de l'afficheur à 7 segments.

Note : Sachant que (par exemple) la représentation binaire du nombre signé "-5" (1011) est le complément à 2 de la représentation binaire du nombre non signé "5" (0101), on pourra se servir de l'additionneur-soustracteur 4 bits réalisé dans le premier laboratoire afin de concevoir ce circuit. Le circuit de l'additionneur-soustracteur 4 bits est fourni sur le site web du laboratoire.

III- Procédure à suivre au laboratoire

III.1 Étape 1 : Afficheur 7 Segments

*N.B.: **N'utilisez jamais** des espaces dans les noms de vos fichiers ou répertoires.*

Créez un nouveau design et en ajoutez un schéma appelé "**Afficheur7Segments**" (ou **A7S**) et entrez le circuit correspondant à vos équations logiques réduites trouvées à la section "**II.2**" (travail préparatoire).

Contraintes de design :

- Pour les **sorties**, utilisez un bus à 8 bits avec le nom **"ssd(7:0)"**, et assignez les différents segments selon la numérotation présentée au tableau de vérité précédent (i.e., *ssd(0) correspond au segment 'a', ssd(1) correspond au segment 'b'*, etc.).

- Pour les **entrées**, utilisez un bus à 4 bits, que vous pouvez appeler **"bit(3:0)"**, où bit(3) correspond au bit3, bit(2) correspond au bit2, etc.

- Vous pouvez également subdiviser l'afficheur 7 segments en 7 sous circuits (1 par segment/ équation), puis rassembler le tout.

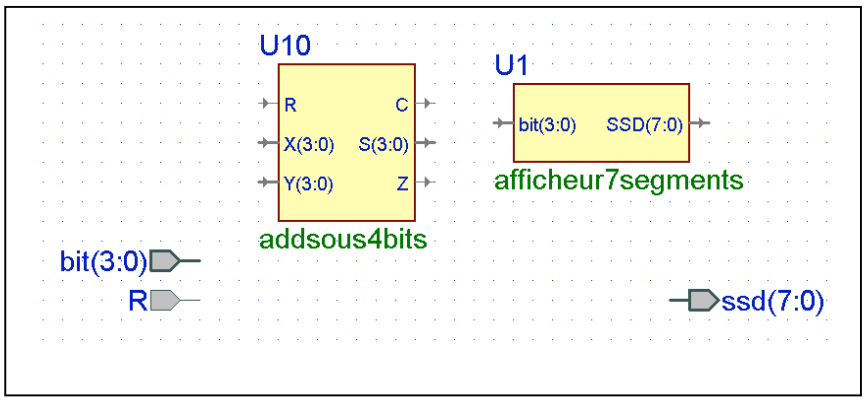
Une fois votre schéma complété, compilez le. Un symbole est automatiquement créé dans l'onglet "Symbol Toolbox". Simulez ensuite le circuit pour vous assurer que les sorties correspondent bien au tableau de vérité.

III.2 Étape 2 : Afficheur complément à 2

Ajoutez le fichier "add1bit.bde" (celui réalisé au TD1) à votre design et compilez . Ajouter le fichier " addSous4bits.bde" (celui réalisé au TD1) à votre design et compilez. Un symbole de l'additionneur-soustracteur 4 bits (addSous4bits) est automatiquement créé dans l'onglet "Symbol Toolbox".

Créez un nouveau schéma appelé "**AfficheurComplement2**" et concevez le circuit permettant de faire l'affichage alterné entre les modes signé (complément à 2) et non signé à partir de l'**Afficheur7Segments** (créé dans III.1), de l'**addSous4bits,** et des portes logiques que vous jugez utiles.

Comme montré dans la figure ci-dessous, ce circuit doit avoir en entrée un bus de 4 bits (**bit(3:0)**) qui représente le nombre à afficher et un signal de sélection (**R**) permettant de sélectionner le type de représentation (**signé si R = 1 et non signé si R = 0**). La sortie de ce circuit reste toujours un bus de 8 bits (**ssd(7:0)**) où chaque bit correspond à un segment de l'afficheur à 7 segments (consulter II.2.2).



Du fichier UCF, enlevez les '#' devant les lignes correspondant aux commutateurs SW (Entrées) et aux signaux SSD (Sorties) que vous utilisez.

Synthétisez (*N'oubliez pas de spécifier le synthétiseur, l'outil d'implémentation et le type de FPGA dans la fenêtre "Design flow → Flow Settings"*. Implémentez, programmez et testez sur la planchette votre circuit. Vérifiez que l'affichage fonctionne correctement dans les deux cas (S et NS).

Montrez le fonctionnement de votre circuit au chargé de laboratoire.

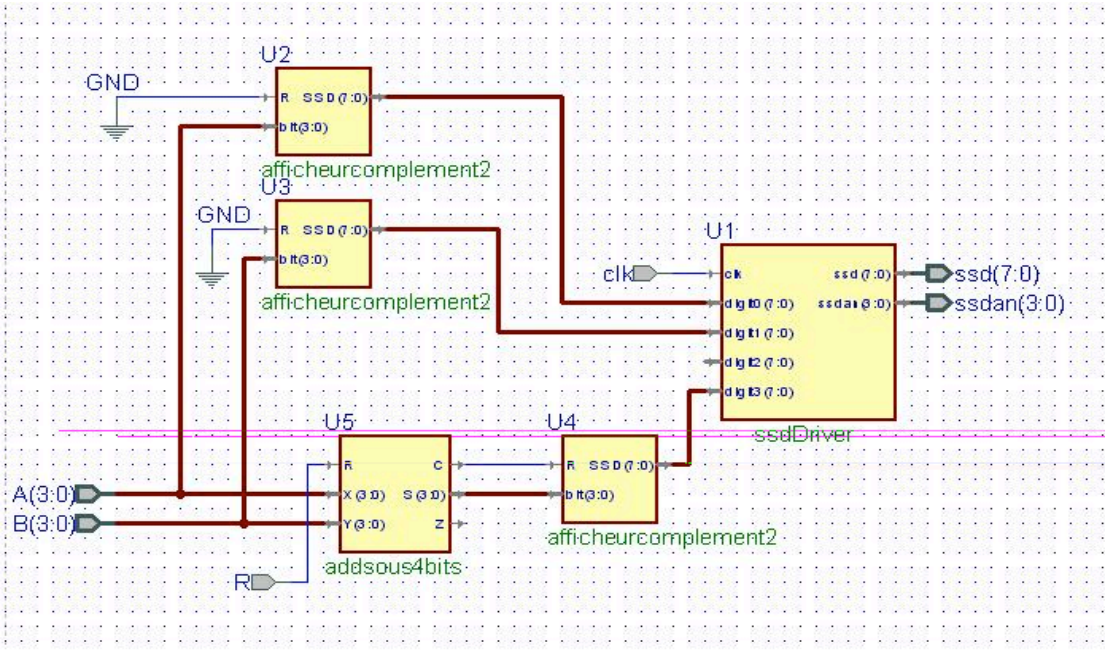
III.3 Application : Calculatrice élémentaire (cette partie vaut 1 point)

Ajoutez une copie du fichier ssdDriver.vhd (voir sur le site du cours) à votre projet. Ce fichier contient un module qui permet d'utiliser les 4 afficheurs ainsi que les 7 segments simultanément. Il fonctionne en acceptant les codes pour quatre chiffres différents, pour les appliquer à tour de rôle sur chaque afficheur individuel. Vous n'avez pas à en comprendre le fonctionnement interne. En compilant le fichier *ssdDriver.vhd*, un symbole sera automatiquement créé.

Créez un nouveau schéma appelé **"SystemeAffichage"** et ajoutez le symbole de "ssdDriver" à celui-ci. Utilisez le design de l'additionneur-soustracteur 4 bits (**addSous4bits**) pour réaliser une calculatrice élémentaire permettant d'effectuer l'addition et/ou la soustraction de deux nombres **hexadécimaux non signés** à 4 bits, comme montré dans la figure suivante. Ajoutez:

- Les entrées A(3:0) et B(3:0) pour les opérandes.
- L'entrée R pour choisir l'opération à effectuer: addition (R = '0') ou soustraction (R = '1').
- L'entrée "clk" (pour l'horloge de la planchette).
- Une sortie "ssd(7:0)".
- Une sortie "ssdan(3:0)".
- 3 modules " AfficheurComplement2" réalisés dans l'étape 2 pour les deux opérandes A et B ainsi que le résultat de l'opération.

Connectez l'ensemble comme montré ci-dessous:



Ajustez le fichier UCF selon les entrées/sorties du système (*N'oubliez pas les lignes qui correspondent à l'horloge* ***"clk"*** *ainsi que les* ***ssdan(3:0)***). Synthétisez, implémentez, programmez et testez votre circuit sur la planchette.