Présentation du TP : L'objectif du TP est de mettre en oeuvre la logique programmable et plus précisémment les FPGA. On travaillera en utilisant la saisie de schéma comme outil de description. Le TP s'organisera sous la forme d'un ou plusieurs petits projets à développer au fil du temps.

Pour les TP, on utilisera la ***carte basys2*** de Digilent dotée d'un ***FPGA Xilinx XC3S250E-CP132*** . ***On mémorisera bien cette référence de circuit pour régler les paramètres du logiciel ISE lors de la création des projets successifs.***

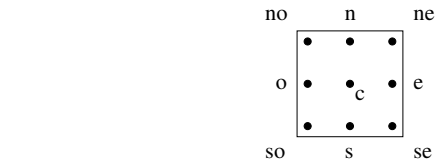
Dans les schémas, les nouveaux composants à programmer dans l'exercice en cours sont dessinés en rose. Les composants déjà programmés dans les exercices précédents ou fournis par l'enseignant sont dessinés en bleu. Le cadre rouge représente le circuit logique programmable. Les signaux entrant ou sortant de ce cadre sont donc les entrées/sorties du FPGA. En général, les noms des signaux sont imposés pour faciliter la correction des erreurs par l'enseignant. Les entrées sont reçues, et les sorties envoyées, sur les broches/bornes/pattes du circuit qui permettront les tests. Là encore, elles sont imposées par le matériel utilisé (carte Basys 2) ou par l'enseignant pour les mêmes raisons de facilité de correction.

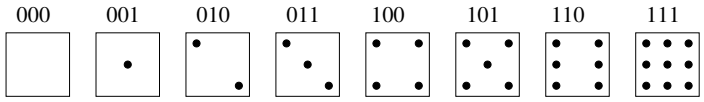
Le premier projet consistera à réaliser un dé électronique.

Exercice 0 : prise en main de l'outil logiciel On réalise ici une simple porte ***et à 2 entrées*** pour bien comprendre ce qui est nécessaire et ce qui se passe lorsqu'on souhaite mettre en œuvre de la logique programmable.

Exercice 1 : transcodeur dé (logique combinatoire) Le dé est constitué de 9 leds formant une matrice 3x3. Cette matrice permettra, bien sûr, de représenter le résultat d'un "jet" du dé avec ses 7 points classiques. Mais, pour obtenir un dé plus complet, on dessinera aussi à terme, des motifs qui représenteront le "temps de roulage" du dé pendant son jet. On commence donc par une version simplifiée des motifs à dessiner selon la figure ci-dessous. Les leds sont nommées c, n, s, e, o, no, ne, so et se pour Centre, Nord, Sud, Est, Ouest, Nord-Ouest, Nord-Est, Sud-Ouest et Sud-Est

Préparation : remplir le tableau de correspondance donnant, pour chaque valeur du dé, le code binaire dessinant le motif souhaité selon la figure ci-dessous en choisissant d'allumer une led avec un '1' logique. Extraire les équations logiques combinatoires de chacune des leds. On remarquera la symétrie de fonctionnement de certaines leds qui permet la recherche d'un nombre réduit d'équations (n et s , e et o, no et se , ne et so).



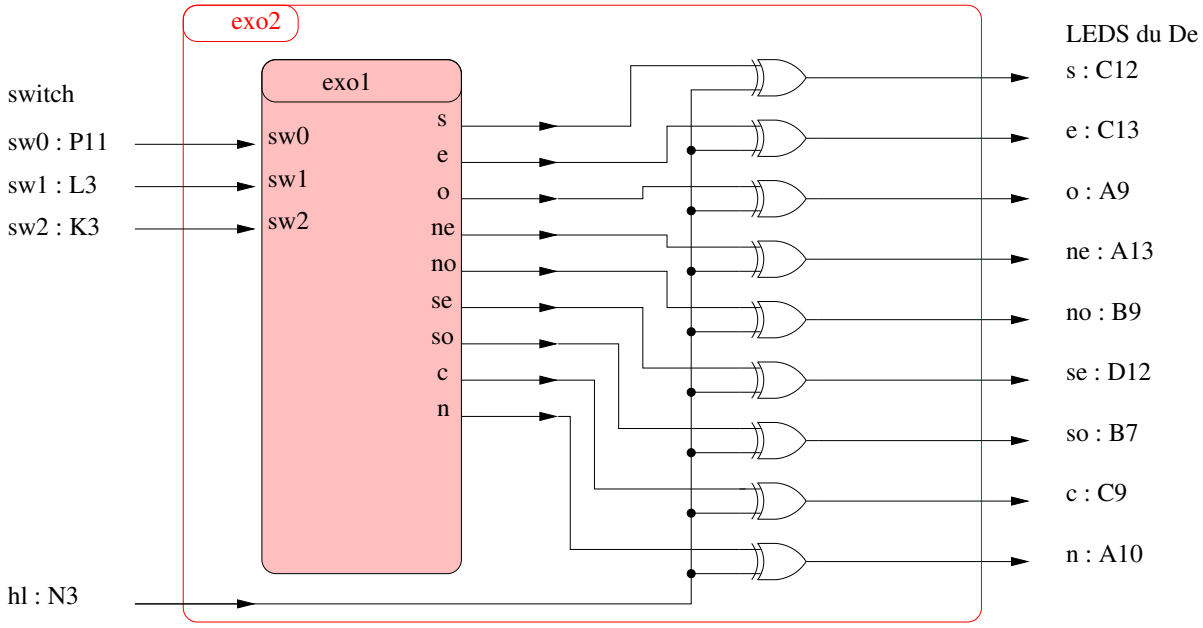


nbre	n	s	e	o	c	ne	no	se	so
000									
001									
010									
011									
100									
101									
110									
111									

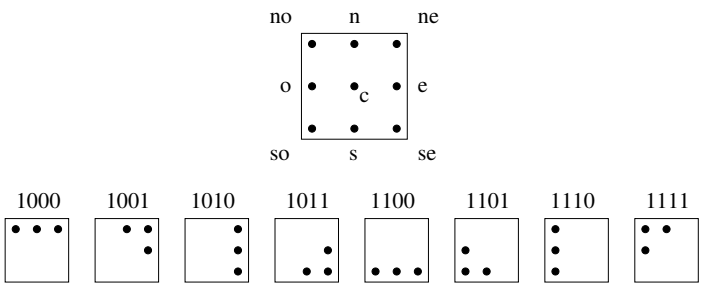
TP : Implanter sous la forme de schéma ***et-ou***, éventuellement simplifié, les équations ainsi extraites. Pour les sorties ayant la même équation, on intercalera un ***buf*** avant le marqueur de sortie pour éviter au logiciel de devoir nommer différemment 2 fois le même signal. Tester et indiquer quel est le défaut de fonctionnement.

Exercice 2 : transcodeur dé commande hl L'exercice 1 présentant un défaut de fonctionnement, on va corriger celui-ci en ajoutant une entrée ***hl*** permettant de commander les leds soit avec des niveaux hauts ('1' logiques) pour une configuration cathode commune, soit avec des niveaux bas ('0' logiques) pour une configuration anode commune.

TP : Préparer un nouveau projet sous ISE. Copier les fichiers ***exo1.sch*** et ***exo1.ucf*** de l'exercice 1 dans le répertoire du projet exercice 2. Renommer ***exo1.ucf*** en ***exo2.ucf***. Ouvrir et ajouter au projet le fichier ***exo1.sch*** puis le sélectionner et utiliser l'outil ***create schematic symbol*** dans ***design utilities*** pour générer un composant ***exo1*** disponible en librairie.
Ouvrir un nouveau fichier ***exo2.sch***, implanter le composant ***exo1*** et ajouter les fonctions logiques selon le schéma de la figure ci-dessous. Compiler; tester et faire valider par l'enseignant.

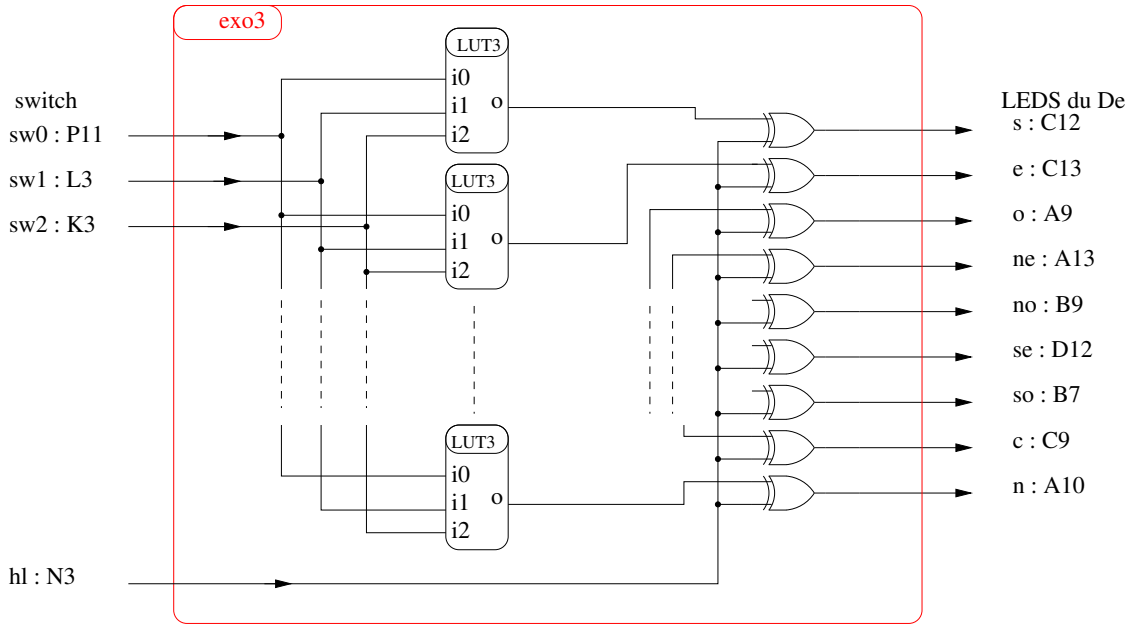


Exercice 3 : transcodeur motifs On souhaite désormais ajouter le codage des motifs représentant le temps de roulage du dé. Le codage est donc étendu à un 4ième bit. Quand celui-ci est à 0, le codage utilisé est celui de l'exercice 2 et on obtient l'affichage des points du dé. En revanche, lorsque ce 4ième bit est à 1, le codage doit être celui de la figure ci-dessous.



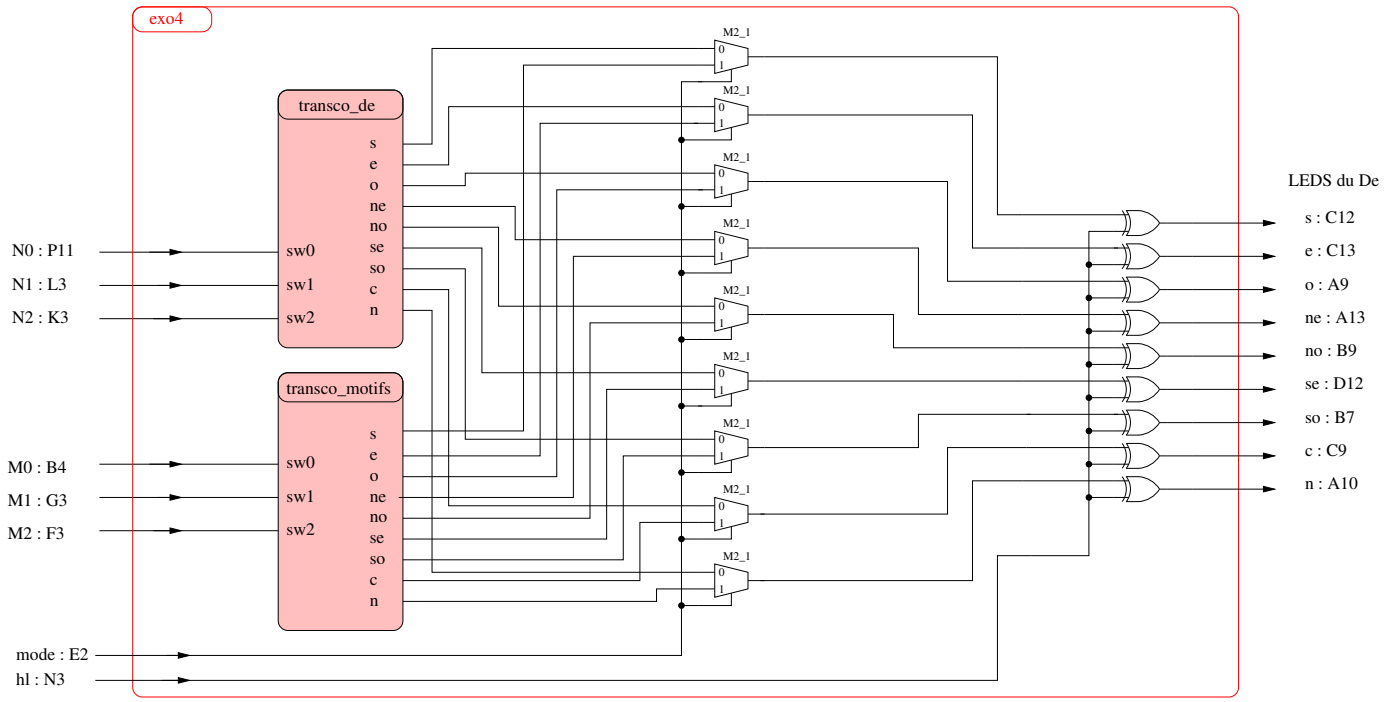
nbre	n	s	e	o	c	ne	no	se	so
000									
001									
010									
011									
100									
101									
110									
111									

TP : Remplir le tableau de correspondance ci-dessus. Implantez les fonctions en utilisant 9 LUT3 selon le schéma ci-dessous. On gardera le principe de fonctionnement de l'exercice 2 exploitant le signal ***hl*** pour gérer les câblages anode et cathode commune.



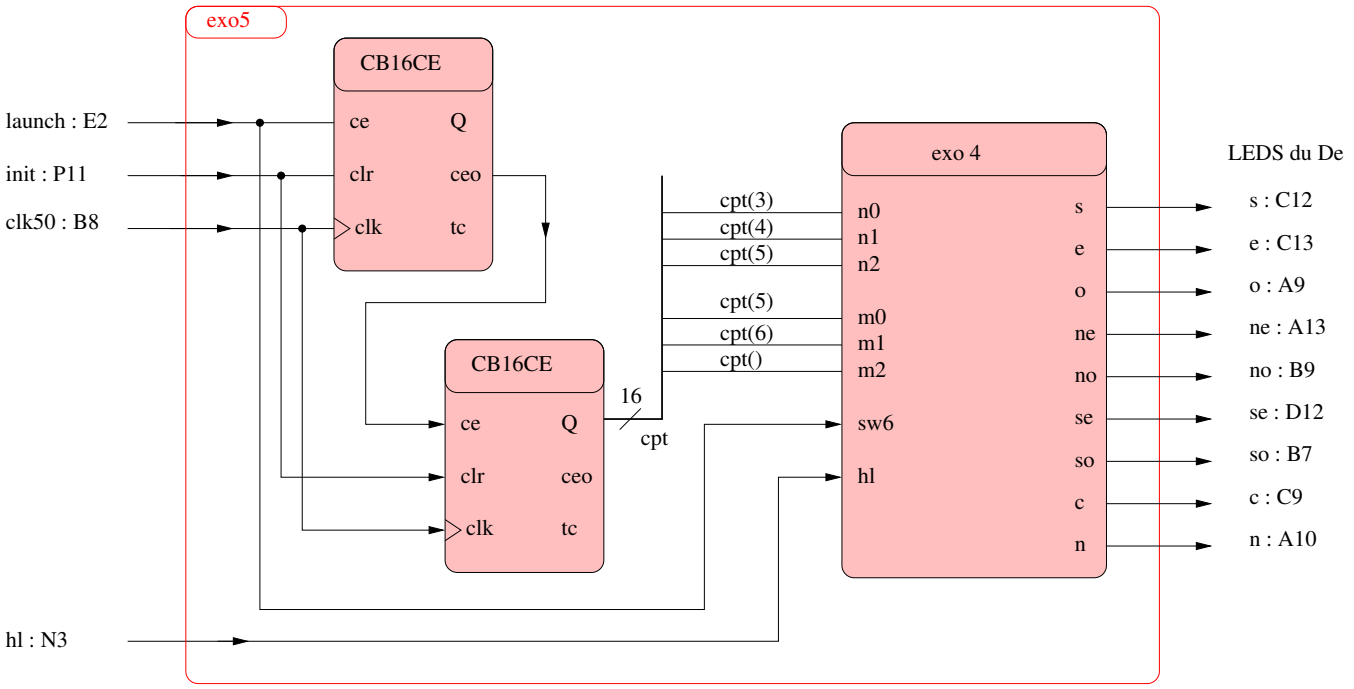
Exercice 4 : transcodeur complet On souhaite désormais implanter le transcodeur complet du dé rassemblant les deux modes de fonctionnement. La mise en oeuvre se fera en multiplexant l'affichage - soit du dé, soit des motifs - avant de le transmettre aux inverseurs commandés (gestion anodes/cathodes communes).

TP : Préparer un nouveau projet ***exo4***. Y recopier les fichiers ***exo3.sch*** , ***exo3.ucf*** et ***exo1.sch*** . Renommer ***exo3.ucf*** en ***exo4.ucf***. Renommer ***exo1.sch*** en ***transco_de.sch*** , Renommer ***exo3.sch*** en ***transco_motifs.sch*** . Ajouter au projet les fichiers exo4.ucf, transco_de.sch et transco_motif.sch. Ouvrir ce dernier schéma et supprimer tous les inverseurs commandés (ou_exclusifs) et l'entrée ***hl.*** Créer alors le symbole associé ***transco_motifs.sym*** . Ouvrir transco_de.sch et créer le symbole associé.
Utiliser alors les 2 transcodeurs (***transco_de et trasnco_motifs***) selon le schéma de la figure ci-dessous dans le fichier ***exo4.sch***. Une série de 9 multiplexeurs 2 vers 1 (m2_1) permet alors au 4 ième bit cité dans l'exercice 3 de sélectionner le mode de transcodage dé ou motif pour piloter les leds. On conservera les inverseurs commandés. Tester le fonctionnement du dé puis du motif en sélectionnant le mode par le bit sw6.



Exercice 5 : Dé complet On souhaite maintenant mettre en place le dé complet pour cela on ajoute 2 compteurs 16 bits (CB16CE) qui vont diviser la fréquence de l'horloge de référence de 50MHz. On utilisera 3 bits du compteur (cpt(19:21)) pour piloter le nombre tiré (***n***)au sort et 3 autres bits (cpt(21:23)) pour piloter le motif affiché (***m***). Le signal launch (***ce*** des compteurs) sur un bouton poussoir (enfoncé) permettra de faire évoluer le compteur et en même temps de sélectionner le mode motif . On verra ainsi le motif défiler. En relâchant ce bouton poussoir, on bloquera le compteur sur une valeur au hasard et l'affichage basculera par la même occasion sur le mode dé. On aura donc l'affichage de la valeur du dé tirée. C'est donc l'instant où on relâche le bouton poussoir qui constitue le tirage aléatoire.

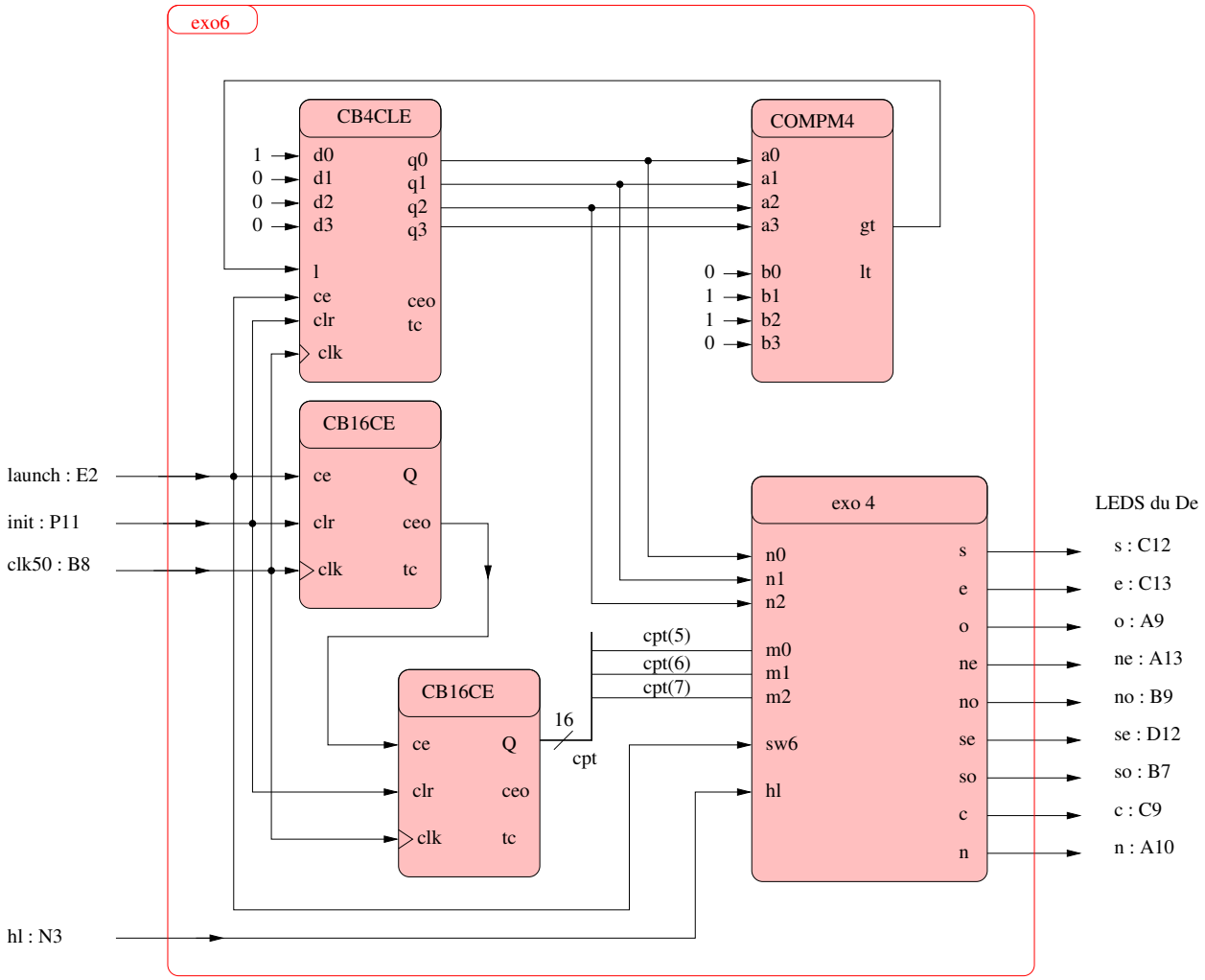
TP : Reprendre tous les fichiers sources de l'exercice 4 et les ajouter à ce nouveau projet exo5. Générer le symbole ***exo4.sym*** à partir de ***exo4.sch*** . Implanter le schéma ci-dessous en exploitant les component des exercices précédents . Tester . Quels sont les petits dysfonctionnement ?



Exercice 6 : Dé corrigé Le problème de fonctionnement réside dans le fait que nous n'avons besoin que des valeurs 1 à 6 dans le tirage au sort de la valeur du dé. Comme on a mis en oeuvre un compteur, sur 3 bits, le transcodage

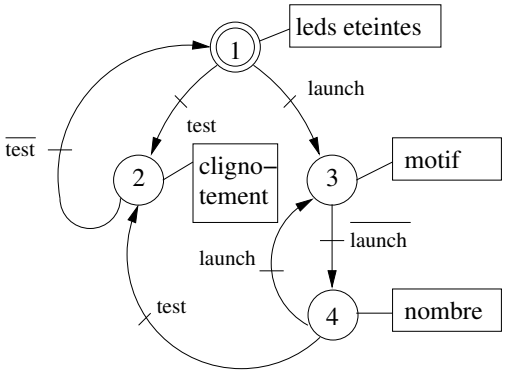
correspond à 8 valeurs (0 à 7). On a opté pour toutes les leds éteintes si la valeur vaut 0 et toutes les des allumées si la valeur vaut 7 . Ceci est judicieux dans le sens où on pourrait ainsi mettre en place un mode de test du dé et ainsi vérifier que toutes les leds fonctionnent a priori. Sinon, on risquerait d'avoir un dé "pipé" ! Le problème apparaît donc de temps en temps quand le tirage aléatoire donne 0 ou 7 comme résultat .
Pour régler ce problème, il suffit de mettre en place un compteur qui balaie uniquement les valeurs 1 à 6. On utillise donc un compteur spécifique CB4CLE (sur 4 bits car on n'a pas de compteur sur 3 bits en librairie). Celui-ci , bien choisi, dispose d'une entrée de pré-chargement. Il suffit donc de forcer le compteur à 1 dès qu'il dépasse 6. On aura ce signal de forçage à 1 en utilisant un comparateur (COMPM4) observant la valeur du compteur par rapport à la valeur fixe 6. Si le signal clr est bien asynchrone, le compteur s'arrêtera bien à 6 pour reboucler à 1. Il restera le cas du 0 mais celui-ci ne se présentera qu'une fois à la mise sous tension ou bien lors d'un reset du système, ce qui ne pose donc pas de problème.

TP : Générer le projet exo6 et y placer tous les fichiers source (.sch , .sym et .ucf) du projet exo5. Renommer exo5.sch et exo5.ucf en exo6.sch et exo6.ucf. Ajouter tous ces fichiers au projet et modifier exo6.sch selon le schéma de la figure ci-dessous. Tester et constater que les cas "toutes leds éteintes" et "toutes led allumées" ne semblent plus se présenter.



Exercice 7 : Dé version FSM Le fonctionnement du dé est correct. On peut chercher à l'améliorer encore en lui donnant d'autres fonctionnalités. Par exemple on peut intégrer un mode de test du dé. Et ainsi vérifier que toutes les leds fonctionnent. On pourrait aussi envisager une version avec affichage numérique de la valeur tirée. Et ainsi accéder à un dé dont la valeur dépasse 6 comme certains jeux de société l'impose (jeux de rôle). On pourrait aussi envisager un dé qui mettrait en œuvre un tirage au sort reposant sur un générateur pseudo-aléatoire. L'imagination est la seule limite... Pour accéder à un dé plus complet, la nécessité de mettre en place une machine d'état devient pressante. En effet, il devient difficile de compléter le dé tel qu'il existe en ajoutant toujours des fonctions autour de ce qui a résulté des exercices précédents. Une machine d'état permet de structurer la mise en oeuvre d'un système dont la complexité augmente.

TP : implanter la machine d'état suivante et la tester.



leds eteintes : 0 pour nombre
clignotement : 0 puis 7 (horloge 3Hz) pour nombre
motif : issu du compteur specifique sur transcodeur
nombre : issu du compteur 1 a 6 pour nombre

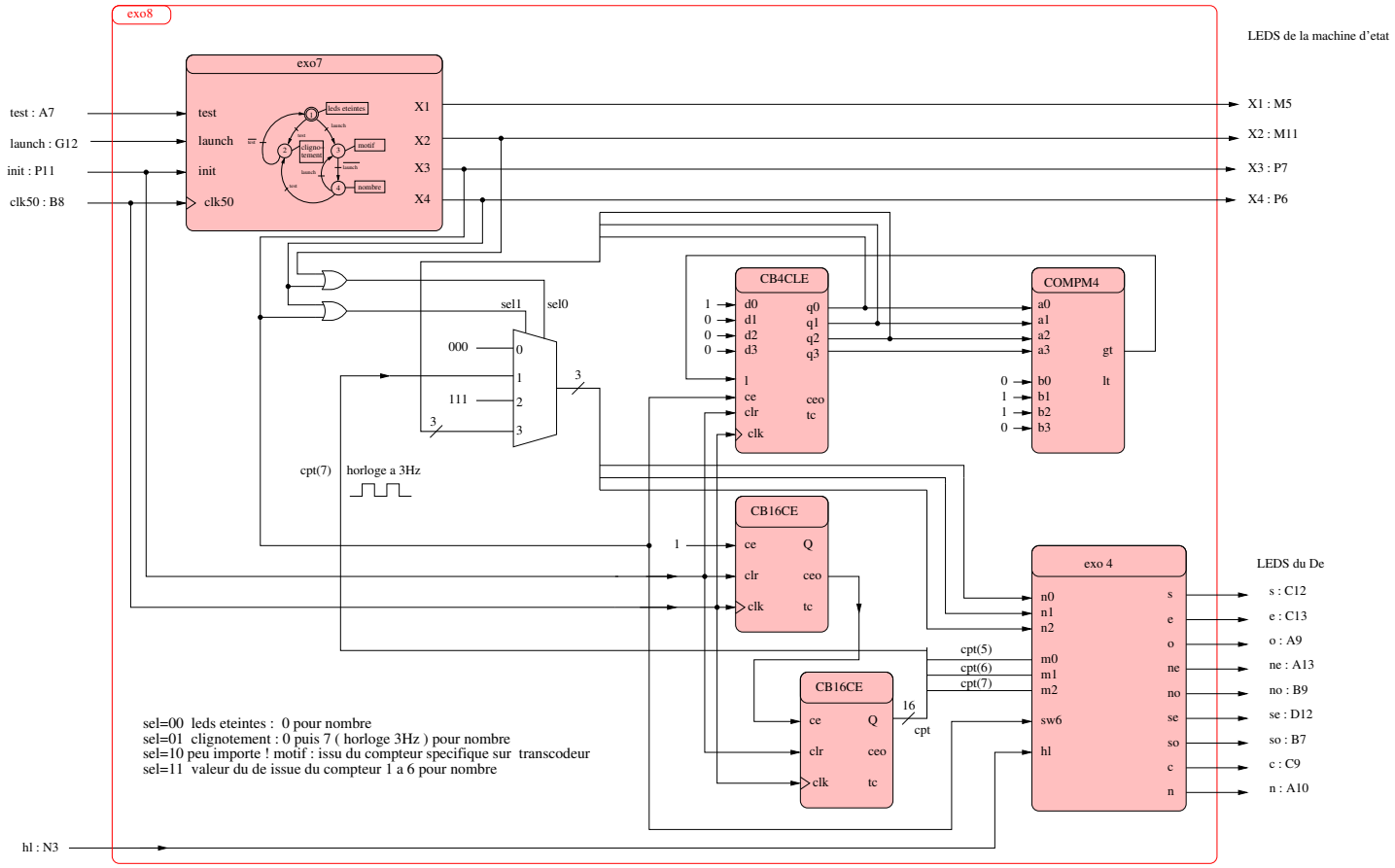
Exercice 8 : Intégration de la machine d'état au dé On ajoute maintenant la machine d'état de l'exercice 7 à l'exercice 6. Il faut pour cela insérer une fonction de multiplexage pour envoyer sur l'entrée nombre (n de exo4) diverses signaux en fonction de l'état de la machine. Ce multiplexeur permet d'envoyer :

- toutes les leds éteintes si on est dans l'état 1
- toutes les leds clignotantes à quelques hertz si on est dans l'état 2
- toutes les leds allumées si on est dans l'état 3 . De fait, ce ne sera pas ce motif qui sera envoyé via l'entrée n mais les motifs défilants pour le roulage du dé présents sur l'entrée m.
- la valeur du nombre tiré (valeur du compteur dé) si on est dan l'état 4

La fonction de multiplexage travaille donc sur 3 bits et permet d'envoyer 1 valeur parmi 4 . On ajoutera donc ***3 multiplexeurs 4 vers 1 (M4_1E)*** en parallèle pour l'implantation du multiplexeur représenté symboliquement sur la figure. De plus, les signaux à modifier seront produits selon les fonctions logiques extraites d'après le tableau ci-dessous. Ainsi $sel1 = X3 + X4$, $sel0 = X2 + X4$, $sw6 = X3$, $CE = X3$. On est donc amené à changer le signal qui alimente l'entrée ***n*** de exo4 ainsi que le signal ***CE*** du compteur pour le dé (CB4CLE) et le signal ***sw6*** du transcodeur déterminant si c'est ***m*** ou ***n*** qui est transcodé.

Etat	sw6	sel1	sel0	CE
1	n - 0	0	0	0
2	n - 0	0	1	0
3	m - 1	1	0	1
4	n - 0	1	1	0

TP : préparer un nouveau projet exo8. Y placer tous les fichiers sources des projets exo6 et exo7 et les ajouter au projet exo8. Ouvrir exo7.sch et générer le symbole équivalent exo7.sym . Implanter alors l'application complète selon la figure ci-dessous.



Exercice 9 : L'exercie 9 et les suivants selon le temps disponible pourront par exemple explorer :

- la mise en oeuvre du langage VHDL
- la mise en oeuvre d'un affichage de la valeur du dé sur afficheur 7 segments
- ...