Travaux pratiques : fonctions logiques de bases

Nous allons au cours de cette séance étudier et simuler quelques fonctions logiques combinatoires et séquentielles de bases à l'aide du logiciel Max+plus II avec lequel nous nous sommes familiarisés lors d'une séance précédente.
Les composants dont nous allons avoir besoin se trouvent dans le répertoire c:\maxplus2\max2lib\prim.
Nous utiliserons les éléments suivants : fonctions de bases (AND, NAND, OR, NOR, NOT, XOR -ou exclusif- XNOR), entrée (INPUT), sortie (OUTPUT), bascule D (appelée DFF), signaux au niveau logique 1 et 0 (Vcc et GND).

1. Fonctions combinatoires

Rappeler ce que l'on entend par fonction combinatoire en électronique numérique.

1.1. Réalisation d'un testeur de parité

Une méthode classiquement utilisée dans les transmissions de signaux numériques pour vérifier le signal reçu, consiste à déterminer si le nombre de bits au niveau logique 1 (NL1) sur un octet reçu est pair.
On compare ensuite le résultat à un bit, dit de parité, envoyé en plus de l'octet par l'émetteur. Si les deux sont différents, on est sur qu'il y a eu une erreur de transmission, s'ils sont identique, cela ne veut cependant pas dire que la transmission est correcte.

Exemple : l'émetteur envoi l'octet « 0101 1010 » suivi du bit de parité qui doit être dans ce cas au niveau logique 1 (pour une parité paire). Le récepteur compte le nombre de bit au NL1 reçus dans l'octet (ici 4) et en déduit que le bit de parité doit être au NL1. On pourrait également compter le nombre total de bits au NL1 (octet plus parité) et vérifier que ce nombre est systématiquement impair dans le cas d'une transmission correcte.

Déterminer les équations logiques liant la sortie de parité P aux deux entrées A0 et A1 d'un détecteur de parité 2 bits (dans le cas d'une parité paire).

Saisir le schéma correspondant dans une description graphique que l'on nommera DETEC_PAIR.gdf.
Compiler en sélectionnant une simulation fonctionnelle (dans le compilateur, valider **Fonctionnal SNF Extractor** du menu **Processing**).

Créer un fichier de simulation .scf permettant d'étudier tous les cas de figure, puis vérifier le résultat en lançant la simulation.

Proposer le schéma d'un détecteur de parité pour un signal de 4 bits A0, A1, A2 et A3 en entrée.
Saisir le schéma, compiler et vérifier.

2. Fonctions séquentielles

Rappeler ce que l'on entend par fonction séquentielle en électronique numérique.

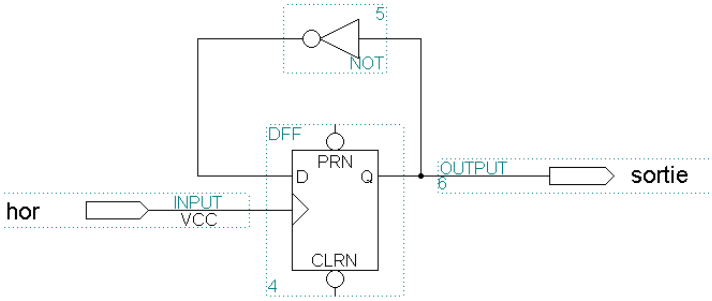
Dans ces fonctions, la mémorisation de l'état des sorties est réalisée par des bascules (nous utiliserons des bascules D). Au front actif d'une horloge de cadencement (un signal carré de fréquence généralement fixe et stable), l'état des sorties change. Le nouvel état dépend de l'état précédent et

des éventuelles entrées
On réalise ainsi un séquenceur, dont les états changent au rythme de l'horloge.

Les fonctions de logique séquentielle sont réalisées à base de bascules câblées en diviseur de fréquence, en compteur ou en séquenceur (un compteur n'est qu'un séquenceur particulier). Nous allons passer en revue ces différents montages de bases.

2.1. Division de fréquence par 2

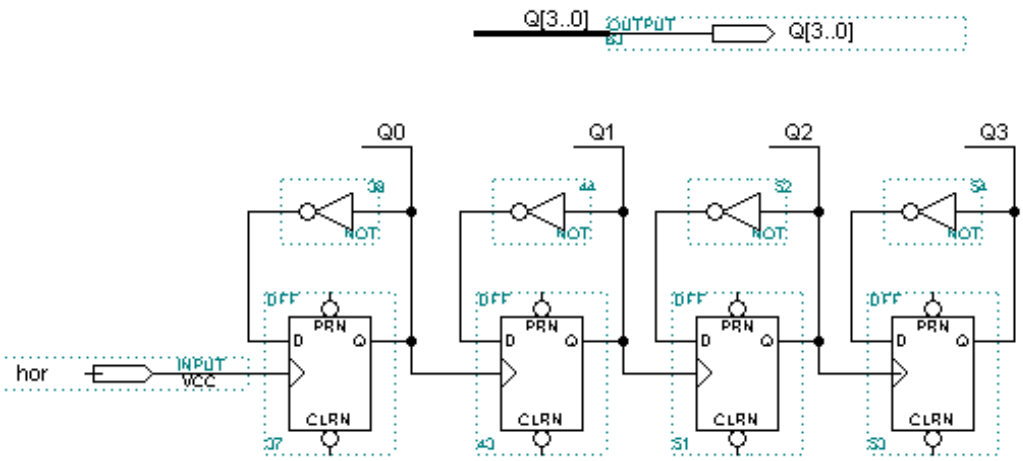
Rappeler le principe de fonctionnement d'une bascule D.
Dans le schéma ci-dessous, si hor est un signal carré, donner l'évolution du signal de sortie.
Quelle rapport existe t-il entre la fréquence du signal hor et celle de la sortie ?



Saisir le schéma et faire une simulation fonctionnelle (l'horloge sera fixée à 20 ns de période par la grille, ce qui évitera d'avoir à la modifier par la suite).

2.2. Compteur diviseur asynchrone

Dans le schéma suivant, déterminer l'évolution des signaux de sortie pour une entrée carrée.



Quel rapport existe t-il entre les fréquences des différents signaux. Déterminer alors la fonction réalisée.
Si on considère que le bus Q[3..0] comprenant les signaux Q3, Q2, Q1 et Q0, représente un nombre binaire, quelle fonction réalise t-on maintenant.

Saisir le schéma dans un fichier COMPT_AS.gdf à partir de la description du diviseur précédent. On utilisera, après avoir sélectionné la cellule de base (clic gauche en restant appuyé et en entourant l'objet souhaité) les fonctions copier (**Ctrl C**) coller (**Ctrl V**) ou bien un "glissé" avec la touche Shift enfoncée, pour multiplier les cellules élémentaires.
Pour donner un nom à une connexion, il suffit de sélectionner la ligne et d'entrer le nom au clavier.
Pour créer le bus Q[3..0], sélectionner le style de ligne épais (clic droit **Line Style**).
Compiler le projet pour une simulation fonctionnelle.

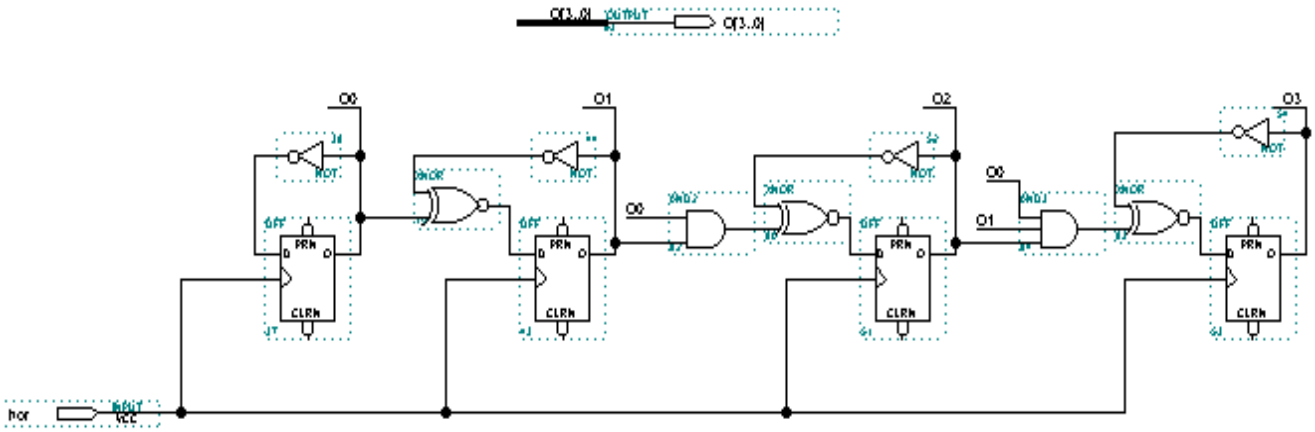
Lors de la description du fichier de test COMPT_AS.scf on introduira également le bus SEG dans les signaux de sortie à visualiser (clic droit **Insert Node from SNF** puis **List** en validant le type **Group**), ce qui permettra de lire directement la valeur sur le bus (en hexadécimal).

Reprendre la compilation, mais cette fois avec une simulation temporelle (hor de période 20 ns). Que remarque t-on ? Conclusion ?

Pour réaliser un compteur, il suffit d'avoir des bascules déclenchant sur front descendant. Pour cela on placera un inverseur sur l'entrée d'horloge. Modifier le schéma et tester.

2.3. Compteur synchrone

Le montage précédent est dit "asynchrone", les sorties n'étant pas synchronisées par rapport à une référence qu'est l'horloge. Comme nous l'avons vu, cela peut entraîner des erreurs importantes au niveau de l'interprétation des résultats. Ce qui est vrai pour un compteur reste vrai pour n'importe quel système séquentiel.
Aussi pour pallier ce problème, on préfère synchroniser le basculement des sorties sur un front d'horloge. On parle alors de systèmes synchrones. Ils sont réalisés en envoyant une horloge commune à toutes les bascules. Les sorties de celle-ci basculent alors toutes au même moment, au prix cependant d'une complexité plus grande du schéma.
On se propose, pour illustrer ce propos d'étudier le montage suivant :



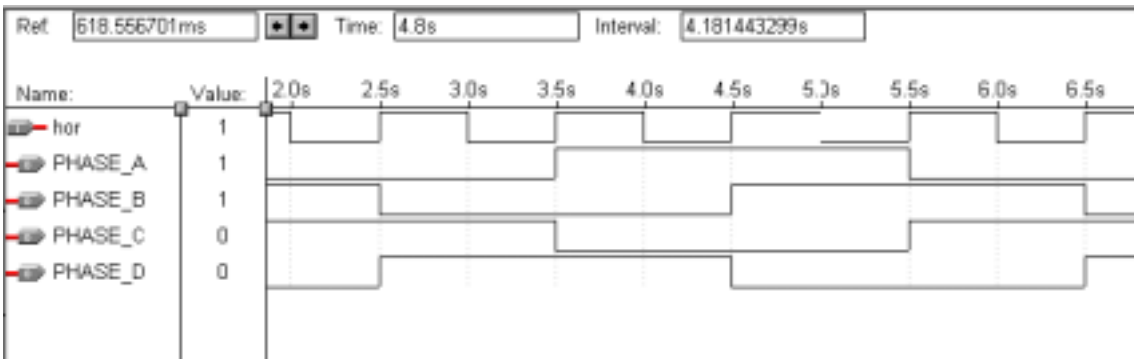
Déterminer l'allure des sorties si hor est un signal carré.
Quel est la fonction réalisée ?
Saisir le schéma, compiler et effectuer une simulation fonctionnelle puis temporelle.

3. Etude d'un séquenceur pour commande de moteur pas à pas

On souhaite maintenant commander un moteur pas à pas 4 phases unipolaire en mode demi-pas (c'est à dire en alimentant toujours deux phases adjacentes à la fois).
Pour tous renseignements complémentaires sur les moteurs pas à pas on se référera au polycopié de cours.

3.1. Etude du séquenceur

La figure suivante donne les chronogrammes attendus à la sortie du séquenceur générant les signaux :



En remarquant que les signaux de sortie sont complémentaires deux à deux, combien faudra t-il de bascules pour synthétiser ce fonctionnement ?
On souhaite un système synchrone.
Quelle relation existe t-il entre la sortie de la première bascule avant le front d'horloge, et la sortie de la seconde bascule après le front d'horloge ? En déduire quel signal relier à l'entrée D de la seconde bascule.
Quelle relation existe t-il entre la sortie de la seconde bascule avant le front d'horloge, et l'entrée de la première bascule après le front d'horloge ? En déduire quel signal relier à l'entrée D de la première bascule.

A partir des éléments précédents, établir le schéma structurel du montage.
Saisir ce schéma dans un fichier .gdf et effectuer une simulation.

3.2. Projet complet

Nous allons maintenant utiliser la carte de développement pour vérifier le fonctionnement de notre système.
Pour représenter le moteur pas à pas, nous utiliserons 4 DEL d'un afficheur de la carte (voir la documentation de la carte pour la position des DEL), une phase étant représentée par une DEL.
Le signal HORLOGE sera obtenu à partir du signal de l'oscillateur à quartz à 25,175 MHz de la carte. On inclura une division de fréquence dans le projet, afin que notre "moteur" tourne à une vitesse d'environ 0,6 tour/s. On pourra pour cela utiliser le sous-ensemble 4count dans la bibliothèque c:\maxplus2\max2lib\mf (pour obtenir de la documentation sur cet ensemble, cliquer sur l'icône "?" puis sur le composant).

Réaliser le projet complet à partir de l'éditeur de schéma et le tester.

Modifier le schéma afin de diviser la vitesse par 4.

Modifier le séquenceur afin d'introduire une entrée SENS définissant le sens de rotation suivant son niveau logique.

Modifier le schéma pour inverser l'entrée SENS tous les quatre tours.