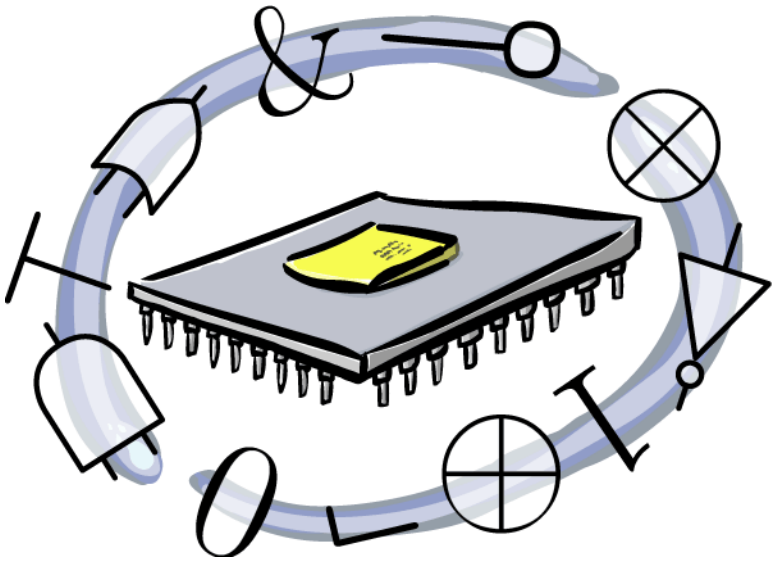
Institut Universitaire de Technologie de CRETEIL-VITRY
Département de Génie Électrique et Informatique Industrielle

SIN1 - SYSTEME D'INFORMATION NUMERIQUE

TRAVAUX DIRIGES



Année universitaire 2013-2014

J. VILLEMEJANE - julien.villemejane@u-pec.fr

GEII - **SIN1**

TD 1

Codage des entiers naturels

1. Code binaire naturel

1. Quel est le nombre minimal de bits p que l'on doit utiliser pour coder un nombre n en binaire naturel (base 2) ? (donner une relation entre p et n)
2. Utiliser la relation précédente pour déterminer le nombre de bits nécessaires à la représentation de 432 en binaire naturel.
3. Rechercher la représentation binaire de ce nombre 432_{10}. Puis donner sa représentation hexadécimale.
4. Convertir les nombres $1001\ 1010_2$ et $10\ 1011\ 0101_2$ codés en binaire naturel en valeur décimale.
5. Donner l'écriture de 193_{10} et de 570_{10}.

2. Addition et soustraction en code binaire naturel

Donner les tables d'addition et de soustraction binaires.
Poser et effectuer, en code binaire naturel, les opérations suivantes :

1. $193_{10} + 570_{10}$
2. $570_{10} - 193_{10}$

3. Multiplication en code binaire naturel *

Donner la table de multiplication binaire.
Poser et effectuer l'opération $13_{10} \times 15_{10}$ en binaire naturel. Puis convertir le nombre binaire obtenu en valeur décimale.

GEII - **SIN1**

TD 2

Codage des entiers relatifs - Langages informatiques

1. Codage des entiers relatifs

1. Coder en code "complément à 1" (C1) puis en code "complément à 2" (C2) les nombres -193_{10} et $+2\,007_{10}$. Penser à coder ces nombres sur le **même nombre de bits** et en n'oubliant pas le bit de signe ! !
2. Trouver le code C2 de $-2\,007_{10}$.
3. Poser et effectuer les opérations suivantes en code C2 en n'utilisant que l'addition binaire :
   (a) $2\,007_{10} - 193_{10}$ (pour information : $a - b = a + (-b)$)
   (b) $-2\,007_{10} - 193_{10}$

Vérifier que les opérations effectuées en binaire donnent le bon résultat en valeur décimale.

2. Langages informatiques évolués et conversion

Un nombre entier est codé sur 8 (`char`), 16 (`short`) ou 32 bits (`int`) en programmation informatique. On parle alors de variable et de type de variable.

1. Donner les valeurs minimales et maximales d'une variable de type `char`, de type `short` et de type `int` (binaire naturel).
2. Combien de symbole hexadécimale faut-il pour coder une variable de type `char` ? De type `int` ?
3. A quoi correspond la valeur hexadécimale `0xA5` en binaire puis en décimale ? Quel type de variable peut-on utiliser pour la stocker ?

3. Code BCD *

1. Coder en BCD pondéré 8421 les nombres $1\,994_{10}$ et 209_{10}.
2. Additionner ces deux nombres en utilisant les règles de l'addition binaire. Le résultat est-il juste ?

Formulation d'un problème logique

1. Formulation d'un problème

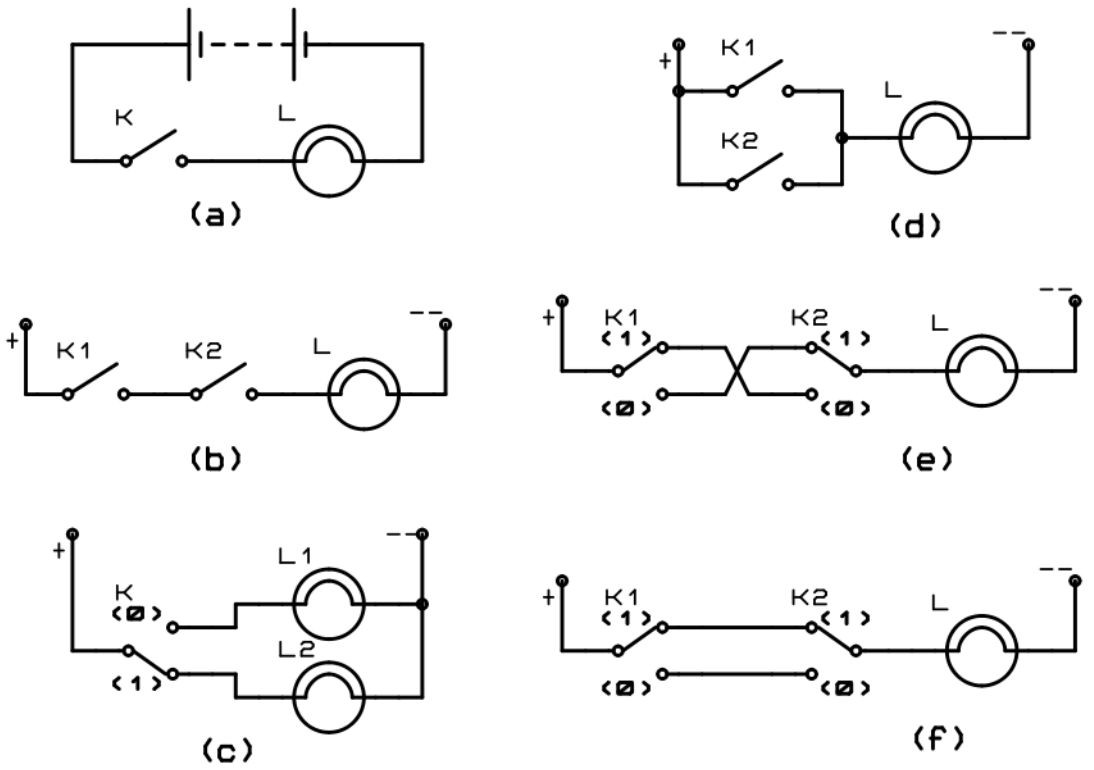
On considère un système composé de 4 lampes dénommées X, Y, Z et S et de 2 interrupteurs nommés I et J. Le système obéit aux lois suivantes :

- la lampe X est éteinte si et seulement si les 2 interrupteurs sont fermés ;
- la lampe Y est éteinte si et seulement si l'interrupteur I est fermé et si la lampe X est allumée ;
- la lampe Z est éteinte si et seulement si l'interrupteur J est fermé et si la lampe X est allumée ;
- la lampe S est éteinte si et seulement si la lampe X et la lampe Y sont allumées.

A quelle(s) condition(s) portant sur les interrupteurs I et J la lampe S sera-t-elle allumée ? Pour résoudre cet exercice, on fera un tableau où l'on portera les "états" des interrupteurs ("O" pour ouvert "F" pour fermé) et des lampes ("E" pour éteint, "A" pour allumé). Il est inutile de faire appel à d'éventuelles connaissances de logique booléenne.

2. Fonctions logiques élémentaires

Dans cet exercice, on étudie la réalisation de fonctions combinatoires simples au moyen d'interrupteurs.



Donner l'équation des différentes lampes L_x en fonction des interrupteurs K_x.
On rappelle que si l'interrupteur est fermé $K = 1$ sinon $K = 0$ et si la lampe s'éclaire $L = 1$ sinon $L = 0$.

GEII - **SIN1**

TD 4

Représentation des systèmes combinatoires

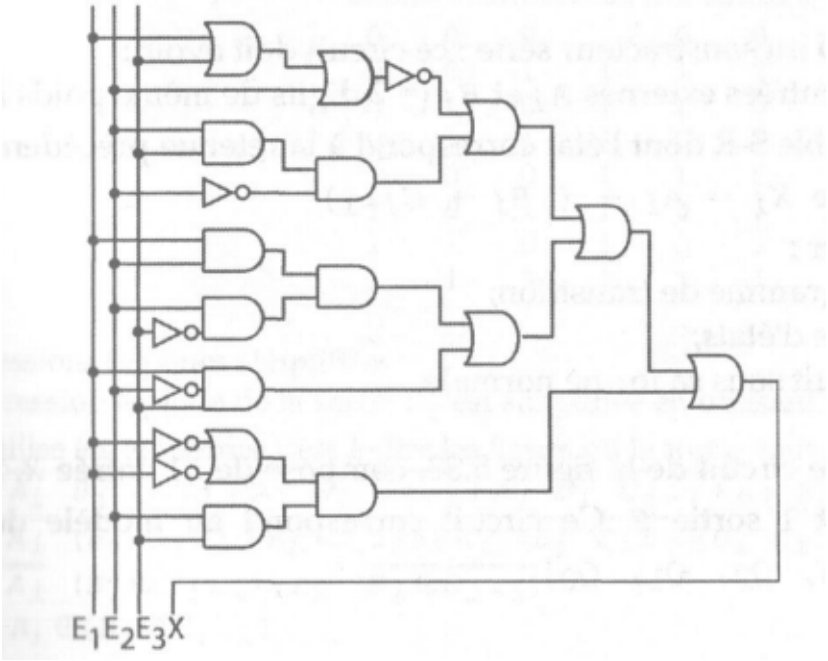
1. Tables de vérité

Faire la table de vérité des expressions suivantes :

- $E_1 = (X + Y) \cdot Z$
- $E_2 = X + (Y \cdot Z)$

2. Analyse d'un circuit logique

Dans cet exercice, on étudie le circuit logique présenté dans la figure suivante.

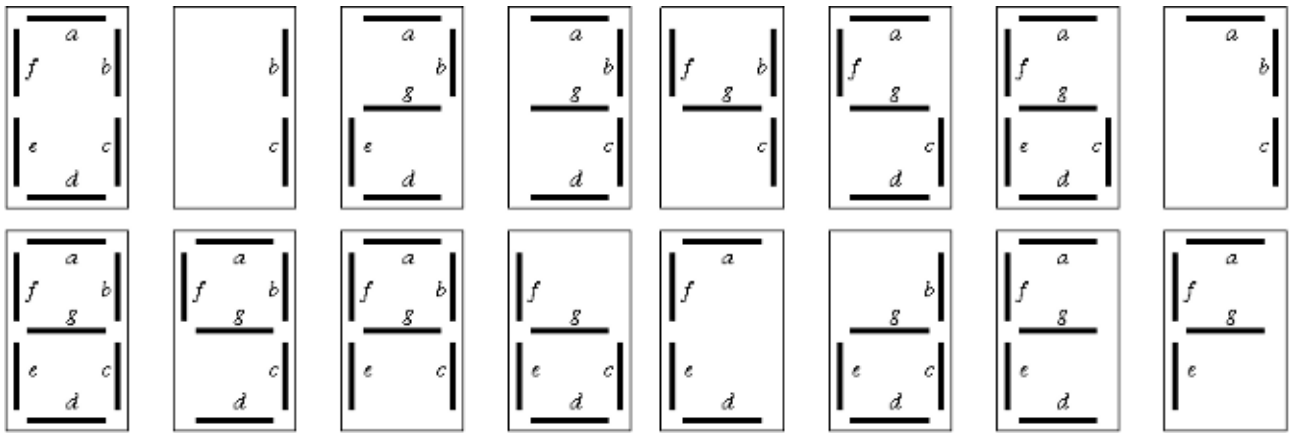


Donner :

1. l'expression booléenne correspondant à la sortie X ;
2. l'expression simplifiée en utilisant uniquement des opérateurs logiques à **deux entrées** choisis parmi les suivants : **ET**, **OU**, **OU-EXCLUSIF**, ainsi que l'opérateur **NON** ;
3. l'expression simplifiée en utilisant uniquement des opérateurs logiques **NON-ET à deux entrées** ;
4. la table de vérité ;
5. le rôle de ce circuit.

3. Afficheur à sept segments *

On désire construire un dispositif de commande d'un afficheur hexadécimal à sept segments. L'afficheur est équipé de 7 sources lumineuses en forme de bâtonnets, commandées par des signaux à 2 états (ou bits) a, b, c, d, e, f, g : lorsqu'un de ces bits à la valeur logique **"0"**, le segment correspondant est allumé. Les symboles sont formés selon le schéma suivant.



Cet affichage devra traduire par un symbole la valeur du nombre formé, en code binaire naturel, par quatre bits D_3, D_2, D_1 et D_0, énoncés ici par poids décroissant.

1. Faire la table de vérité du dispositif de commande : état des segments en fonction de l'état des bits D_3, D_2, D_1 et D_0.
2. On veut simplifier cet afficheur pour le cas où le nombre à afficher ne comporte qu'un seul bit X : si $X = 0$, on doit afficher "0" et si $X = 1$, on doit afficher "1". Faire la nouvelle table de vérité et en déduire les expressions (extrêmement simples) de a, b, c, d, e, f, et g en fonction de X.

GEII - **SIN1**

TD 5

Simplification analytique et graphique d'équations

1. Théorèmes généraux

1. Considérons le théorème du consensus sous sa forme directe : $X \cdot Y + Y \cdot Z + \overline{X} \cdot Z = X \cdot Y + \overline{X} \cdot Z$
   Démontrer *algébriquement* ce théorème.
2. Simplification au moyen des théorèmes généraux : considérer la fonction

$$f(A, B, C, D) = (AD + \overline{A}C)[\overline{B}(C + B\overline{D})]$$

   (a) Dessiner le schéma logique de f en utilisant les symboles "et", "ou", "non".
   (b) En utilisant l'algèbre de Boole, simplifier la fonction et dessiner le schéma résultant.

2. Ecriture de fonctions sous forme canoniques *

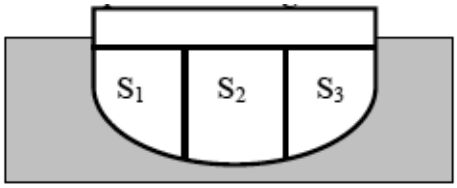
1. Soit la fonction logique $f(A, B, C, D) = AB(C + \overline{D}) + \overline{A + \overline{(B + C)}(\overline{A} + D)}$.
   (a) Représenter sa table de vérité.
   (b) En déduire son écriture canonique en *minterms*.
   (c) Donner son écriture en *maxterms*.
2. Soit la fonction $f(A, B, C, D) = \Sigma m(0, 1, 2, 7, 8, 9, 10, 15)$.
   (a) Réécrire cette fonction sous forme canonique en *maxterms*.
   (b) Ecrire la fonction $\overline{f}$ sous les deux formes canoniques.

3. Simplification de fonctions à l'aide d'un tableau de Karnaugh

1. Déterminer la réalisation minimale en somme de produits des fonctions :
   (a) $f(W, X, Y, Z) = \Pi M(1, 3, 7, 9, 11, 15)$.
   (b) $f(A, B, C, D) = \Sigma m(0, 2, 4, 6)$.
   (c) $f(A, B, C, D) = \Pi M(0, 1, 6, 7)$.
   (d) $f(A, B, C) = A \oplus B \oplus C$.
   (e) $f(A, B, C, D) = \Sigma m(0, 1, 6, 7)$.
2. Simplifier les fonctions suivantes (on utilise la notation $\Sigma d()$ pour les états indifférents) :
   (a) $f(W, X, Y, Z) = \Sigma m(1, 7, 11, 13) + \Sigma d(0, 5, 10, 15)$.
   (b) $f(A, B, C, D) = \Sigma m(1, 3, 5, 7, 9) + \Sigma d(6, 12, 13)$.

4. Soutes d'un navire **

Un navire, destiné au transport d'éléments liquides, comporte dans sa cale trois soutes S_1, S_2 et S_3 (voir schéma ci-dessous). Une soute est soit VIDE ('0'), soit PLEINE ('1').



Le voyant (*AC*) s'allume quand l'assiette est "correcte", c'est-à-dire quand les charges sont bien réparties. Les cas où AC s'allume sont les suivants :

- soutes 1 et 3 vides, soute 2 remplie ;
- soutes 1 et 3 remplies, soute 2 vide ;
- soutes 1, 2 et 3 remplies ;
- soutes 1, 2 et 3 vides.

1. Rechercher la table de vérité correspondant au fonctionnement du voyant AC. On utilisera le binaire réfléchi (Gray) pour le codage des entrées.
2. Rechercher algébriquement l'équation minimale de AC.
3. Représenter AC en utilisant des portes quelconques à 2 entrées.

TD 6

Fonctions combinatoires standard

Codeurs, décodeurs

Tout codeur (ou décodeur) [1] peut se ramener à un dispositif permettant d'associer à un code donné, constitué par un ensemble de n variables logiques, un autre code constitué par un autre ensemble de m variables logiques.

La correspondance n'est pas nécessairement bijective, les nombres de codes *réellement* utilisés par une application étant quelconques.

1.1. Traducteur ASCII → décimal

On veut réaliser un codeur transformant les codes ASCII constitués par 7 variables $a_6 \cdots a_0$ en code binaire naturel lorsque le caractère ASCII est un chiffre. Ce codeur devra fournir 4 signaux $d_3 \cdots d_0$ constituant un code binaire naturel, et une variable de signalisation *decimal*.

Le fonctionnement souhaité est décrit par le tableau ci-dessous :

Caractère	Code ASCII ($a_6 \cdots a_0$)	$d_3 \cdots d_0$	*decimal*
'0'	0110000	0_{10}	1
'1'	0110001	1_{10}	1
'2'	0110010	2_{10}	1
'3'	0110011	3_{10}	1
'4'	0110100	4_{10}	1
'5'	0110101	5_{10}	1
'6'	0110110	6_{10}	1
'7'	0110111	7_{10}	1
'8'	0111000	8_{10}	1
'9'	0111001	9_{10}	1
Autres	autres codes	15_{10}	0

1. Montrer *par le raisonnement* que la fonction *decimal* s'exprime par :

$$decimal = \overline{a_6} \cdot a_5 \cdot a_4 \cdot (\overline{a_3} + (a_3 \cdot \overline{a_2} \cdot \overline{a_1}))$$

2. Simplifier cette équation.
3. Montrer que chaque fonction d_i s'exprime très simplement en fonction de la variable a_i correspondante et de $\overline{decimal}$.
4. Faire le schéma logique du codeur.
5. Donner une description VHDL du codeur (on pourra utiliser un *process*).

1.2. Codeur pour affichage décimal

On considère un dispositif comprenant 4 entrées $D_3 \cdots D_0$ et 10 sorties $L_9 \cdots L_0$ *actives à l'état bas*. Ce décodeur doit activer la sortie dont le numéro correspond au code binaire naturel produit par les bits $D_3 \cdots D_0$.

1. Ecrire les équations des fonctions $L_9 \cdots L_0$.
2. Ecrire diverses descriptions VHDL de ces fonctions.

1. On ne considère ici que des systèmes de codage instantanés, c'est à dire sans mémoire. Il existe des codes où l'interprétation est fonction du contexte.

TD 7

Fonctions combinatoires standard 2

1. Fonctions arithmétiques

1.1. Additionneur / soustracteur en VHDL

VHDL simplifie beaucoup la description de l'additionneur puisque l'addition et la soustraction sont des opérations de base du langage ! La seule chose à laquelle on doit prêter attention est la taille respective des opérandes et du résultat.

Donner la description VHDL d'un additionneur/soustracteur d'entrées $A_3 \cdots A_0$, $B_3 \cdots B_0$, de sortie $\Sigma_3 \cdots \Sigma_0$ et C_o effectuant soit l'addition soit la soustraction de A et B selon l'état d'une variable de commande Op.

1.2. Multiplicateur combinatoire

Le but de cette étude est la réalisation d'un multiplicateur 4 bits x 4 bits, opérant sur des nombres entiers positifs, codés en binaire naturel.

1. Montrer comment on effectue la multiplication $\{A_3 \cdots A_0\} \times \{B_3 \cdots B_0\}$, à la main en faisant apparaître les produits partiels $\{R_3 \cdots R_0\}$, $\{S_3 \cdots S_0\}$, $\{T_3 \cdots T_0\}$ et $\{U_3 \cdots U_0\}$ et le produit final $\{P_7 \cdots P_0\}$.
2. On considère que l'on dispose de trois additionneurs complets à 4 bits et de portes logiques élémentaires en nombre illimité. Faire le schéma d'un multiplicateur à 4 bits suivant l'algorithme précédent.
3. En s'inspirant du travail précédent, donner la description VHDL du même multiplicateur.

2. Comparateur

La fonction de comparaison de deux nombres A et B codés en binaire naturel fournit trois indicateurs à 2 états :

- l'indicateur d'égalité E, affirmé si $A = B$;
- l'indicateur "inférieur" I, affirmé si $A < B$;
- l'indicateur "supérieur" S, affirmé si $A > B$.

Pour concevoir un opérateur de comparaison modulaire, on le décompose en blocs élémentaires de taille un bit, cascadables. Comme pour l'additionneur, on commence d'abord par un opérateur de taille un bit, non cascadable que l'on modifiera ensuite.

2.1. Comparateur à un bit non cascadable

Ce comparateur ne possède que 2 entrées A et B et 3 sorties E, I et S.

Donner les équations et la description VHDL des 3 fonctions E, I et S.

2.2. Comparateur à un bit cascadable

On considère désormais un comparateur pour les bits de rang n de deux nombres binaires A et B, notés A_n et B_n. Ce comparateur est équipé de 3 entrées permettant de prendre en compte le résultat de la comparaison des bits précédents. Ces entrées sont notées E_{n-1}, I_{n-1} et S_{n-1}. Trois sorties sont prévues pour les résultats de la comparaison pour le rang n : E_n, I_n et S_n.

1. A quelle condition portant sur E_{n-1}, A_n et B_n a-t-on $E_n = 1$?
2. En déduire l'équation donnant E_n (l'usage de l'opérateur $\oplus$ simplifie l'écriture).

3. Si l'on suppose que $A_n = B_n$, quelle relation existe-t-il entre I_n et I_{n-1} d'une part, et entre S_n et S_{n-1} d'autre part ?
4. Si l'on suppose que $A_n \neq B_n$, donner les expressions de I_n et S_n.
5. Récapituler les résultats pour E_n, I_n et S_n dans 3 tables de vérité. Ecrire aussi les équations (l'opérateur $\oplus$ simplifie beaucoup l'écriture).
6. Faire le schéma de l'étage n du comparateur, avec les opérateurs "et", "ou", "non", "ou exclusif".

2.3. Comparateur à plusieurs bits

1. Montrer comment on peut assembler plusieurs blocs de comparaison à un bit pour constituer un comparateur multibits.
2. Sachant que VHDL dispose de tous les opérateurs relationnels, donner la description des fonctions E, I et S pour un comparateur à 4 bits.

GEII - **SIN1**

DÉPARTEMENT GÉNIE ÉLECTRIQUE & INFORMATIQUE INDUSTRIELLE

TD 8

Bascules et mémorisation

1. Bascule JK

1. Rappeler la table de fonctionnement d'une bascule JK déclenchée sur front montant.
2. Représenter ces résultats par un diagramme d'état.

2. Bascules D, T et JK synchrones

On dispose d'une bascule D.

1. Concevoir la partie combinatoire à associer à cette bascule pour obtenir le comportement d'une bascule JK (il vous est conseillé d'écrire l'état de la sortie Q_{n+1} en fonction de K_n, J_n et Q_n).
2. De la même façon, concevoir une bascule JK à partir d'une bascule T.

GEII - **SIN1**

TD 9

Synthèse de compteurs

1. Compteur binaire progressif

1. Faire la **synthèse structurelle** d'un **compteur binaire** comptant de **0 à 7** à partir de **bascules JK** synchronisées par front montant.
2. Faire la **synthèse structurelle** du même compteur basé sur des **bascules D**.
3. Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même compteur.

2. Compteur modulo 5

On souhaite à présent concevoir un **compteur modulo 5** synchrone à base de **bascules D**.

1. Tracer les chronogrammes correspondant au fonctionnement du système.
2. Représenter l'évolution du système sous forme d'un diagramme d'états.
3. Ecrire la table des transitions.
4. Représenter la **structure finale** du compteur modulo 5 (équations des entrées D).
5. Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même compteur.

3. Compteur code Gray 3 bits *

En suivant la même procédure que dans l'exercice précédent, réaliser la synthèse d'un **compteur** suivant la séquence d'un **code de Gray de 3 bits**, au moyen de **bascules T**.

TD 10

Synthèse de registres

1. Compteur Johnson

Un compteur Johnson, basé sur des bascules D, doit générer la séquence 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001.

1. Faire la **synthèse structurelle** de ce compteur.
2. Faire son diagramme d'états.
3. Examiner les états non cités jusqu'ici. Y-a-t-il des fonctionnements cycliques indésirables ?
4. Par quel moyen est-on assuré d'obtenir un démarrage correct ? Proposer un schéma simple basé sur des composants électriques passifs.
5. Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même compteur.

2. Registre universel

Les registres à décalage sont normalement utilisés pour décaler les données de manière circulaire (les données sont extraites à une extrémité pour être réinjectées à l'autre), logique (les positions libérées sont remplies par des 0 ou des 1) ou arithmétique (on propage le bit de poids fort vers la droite ou on injecte des zéros vers la gauche).

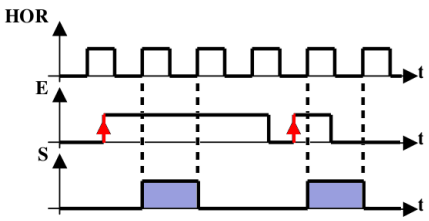
Ainsi, avec un registre de 4 bits, 6 décalages sont possibles :
- décalage circulaire à droite : 1110 → 0111 ;
- décalage circulaire à gauche : 1110 → 1101 ;
- décalage logique à droite : 1110 → 0111 ;
- décalage logique à gauche : 1110 → 1100 ;
- décalage arithmétique à droite : 1110 → 1111 ;
- décalage arithmétique à gauche : 1110 → 1100 ;

Faire le schéma de câblage d'un registre TTL 74xx194 à 4 bits pour obtenir les 6 fonctions précédentes (voir extrait de la notice donné en annexe).

Synthèse d'une machine à état

1. Détecteur de front montant

On considère un signal E susceptible de changer d'état à n'importe quel instant. On désire construire une machine à état synchronisée par une horloge H, produisant un signal S qui devra passer à l'état 1 pendant une période d'horloge après apparition d'un front montant sur E.



Il est rappelé que E peut rester dans le même état pendant un nombre indéfini de périodes d'horloge. Seule la transition de 0 à 1 doit être signalée par une impulsions unique dont la durée est égale à une période d'horloge.

1. Faire le **schéma synoptique** de ce système.
2. Montrer que cette machine devra posséder 3 états et non pas 2, comme on peut le penser à priori.
3. Faire le **diagramme d'état**.
4. Faire la **synthèse structurelle** de ce système basé sur des **bascules D**.
5. Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même système.

2. Identificateur de séquence

On désire concevoir une machine destinée à analyser un flot de bits en série. A chaque front montant d'une horloge de référence (fournie sous forme d'un signal H), apparaît un nouveau bit sur une ligne de donnée (signal D). Le dispositif à concevoir doit reconnaître la séquence de bits 101101 dans le flot. A chaque fois que cette séquence a été identifiée, la machine devra signaler ce fait en plaçant sa sortie S à 1 pendant 1 cycle d'horloge.

1. Faire le **schéma synoptique** de ce système.
2. Tracer le **diagramme d'états** de cette machine.
3. Combien nécessite-t-elle, *au minimum*, de bascules ?

   On décide de synthétiser la machine selon la méthode "*un parmi n*", qui utilise autant de bascules qu'il y a d'états. Cette approche simplifie beaucoup la logique combinatoire, au détriment d'un "gaspillage" de bascules. L'état n de la machine sera codé par la présence d'un 1 dans la bascule de numéro n, toutes les autres bascules étant à 0. Lors du parcours du diagramme d'état, le 1 "se promène" dans les différentes bascules.

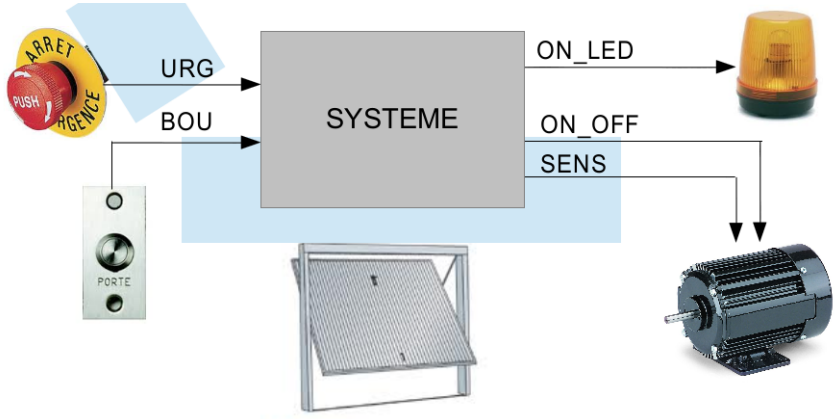
   Les bascules retenues pour cet exercice sont des bascules D, synchrones, déclenchées par front montant.
4. Ecrire la **table des transitions** de la machine.
5. Faire la **synthèse structurelle** de ce système basé sur des **bascules D**.
6. Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même système.

GEII - **SIN1**

TD 13

Automatisme de porte de garage

Une porte de garage motorisée comporte un moteur électrique réversible à 2 sens de marche correspondant à la montée ou à la descente. Il possède 2 entrées : une pour l'alimentation et une seconde pour le sens de rotation.



On se propose de réaliser le système de commande de cette porte motorisée (**machine à état séquentiel synchrone**). Le système est décrit par la suite.

- La commande est assurée par un bouton poussoir unique, **BOU** qui commande la marche : ouverture lorsque la porte est fermée, fermeture lorsque la porte est ouverte.
- Deux capteurs de fin de course haut et bas, **FH** et **FB**, permettent d'arrêter automatiquement le mécanisme.
- Pour représenter la sortie du système, on utilisera un vecteur **CMD** à 2 composantes :
  - **CMD(0)** représente l'activité : marche si CMD(0) = '1', arrêt sinon ;
  - **CMD(1)** représente le sens de marche : montée si CMD(1) = '1', descente sinon.
- Lorsqu'une commande est en cours d'exécution, le bouton **BOU** est inopérant.
- La montée ou la descente seront visualisées par 2 diodes **MONTEE** et **DESCENTE**.

1. Etude du **système sans sécurité** :

   (a) Faire le **diagramme d'état** (3 états A, M et D sont suffisants).

   (b) Faire la **synthèse structurelle** de ce système en utilisant des bascules D.

   (c) Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce même système.

2. Etude d'une **sécurité simple** :
   Un détecteur d'urgence, URG, permet d'arrêter immédiatement le fonctionnement, quel que soit l'état du système, à la position où il se trouve ; dans ce cas, après élimination du défaut ($URG =' 0'$), un appui sur BOU permet de reprendre le mouvement dans le sens précédent.

   (a) Faire le nouveau **diagramme d'état**.

   (b) Faire la **synthèse comportementale**, basée sur le langage VHDL, de ce nouveau système.

GEII - **SIN1**

TD Bonus

Logique de décodage d'un décodeur en quadrature

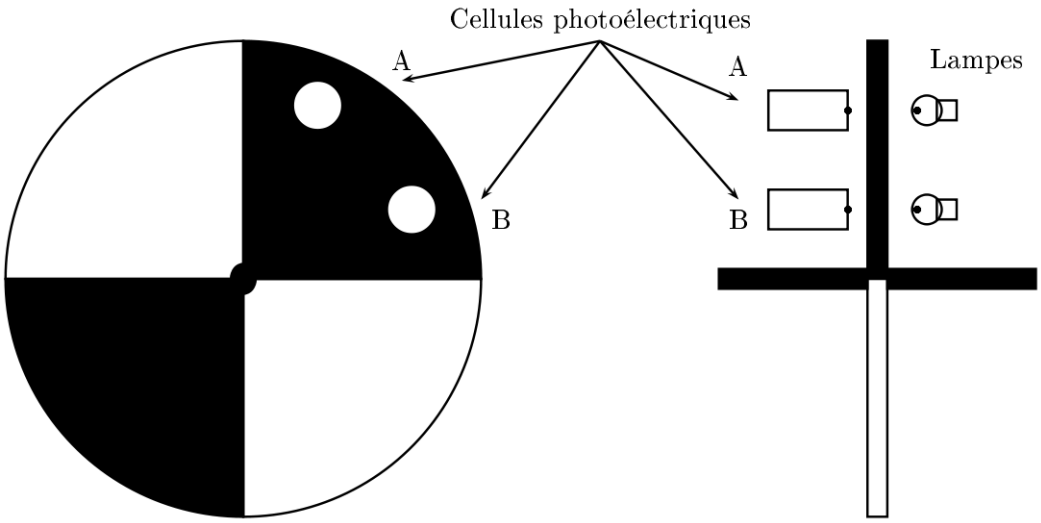
Pour connaître la position d'un axe de machine tournante, on l'équipe d'un **codeur en quadrature**, selon le schéma de la figure ci-après. Dans ce montage, une roue à 2 dents (en noir sur la figure) est solidaire de l'arbre de la machine tournante. Les dents sont des secteurs de 90 degrés opaques, séparés par un évidement. Deux couples lampe/cellule photoélectrique sont disposés à égale distance de l'arbre, de telle manière que le faisceau de la lampe puisse être interrompu par les dents de la roue. Les 2 ensembles lampe/cellule sont fixes et forment avec l'axe un angle de 45 degrés. Chaque cellule photoélectrique est reliée à un circuit d'adaptation dont la sortie (respectivement A ou B) est un niveau logique "1" lorsque la cellule *n'est pas éclairée* et un niveau logique "0" dans le cas contraire.

On négligera le temps pendant lequel une cellule est partiellement éclairée.

Les deux signaux A et B seront appliqués à l'entrée d'une machine à état synchrone, dont l'horloge H a une fréquence considérée comme très élevée par rapport à la fréquence maximale de A et B. Le sens de rotation peut être déterminé à partir de l'examen des séquences de signaux A et Bd.

Le principe du codage incrémental est d'incrémenter (de un !) un compteur lorsque l'axe tourne dans le sens positif, de le décrémenter (de un !) lorsqu'il tourne dans le sens négatif, et de le laisser inchangé lorsque l'axe ne tourne pas.

Le but de la machine à état à concevoir est de fournir au compteur l'indication de l'opération à effectuer : incrémenter (+1), décrémenter (-1) et ne pas changer (0).



Conventions et notations

On appelle A_n et B_n les valeurs logiques de A et B à la nième période d'horloge et A_{n-1}, B_{n-1} leurs valeurs à la période immédiatement précédente.

Le codage des opérations du compteur nécessite 2 bits P et Q selon le code suivant :

- $P = 0$, $Q = X$ pour "ne pas changer" ;
- $P = 1$, $Q = 1$ pour "incrémenter" ;
- $P = 1$, $Q = 0$ pour "décrémenter".

Questions

1. Faire un diagramme des successions d'états de A et B, pour chaque sens de rotation.
2. Faire figurer sur ce diagramme les "transitions" correspondant à l'immobilité.
3. Trouver, mais ne pas les porter sur le diagramme, les transitions impossibles.
4. Remplir le tableau ci-dessous :

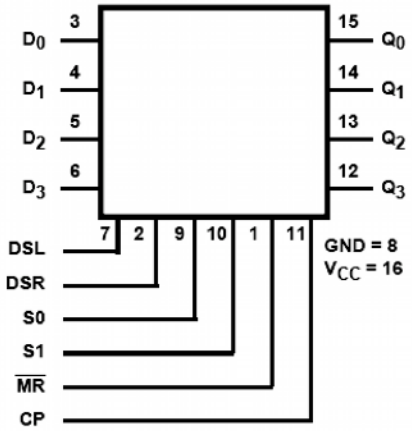
A_{n-1}	B_{n-1}	A_n	B_n	P	Q
0	0	0	0	0	X
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
etc.					

5. On choisit de prendre A et B comme variables d'état de la machine : faire le synoptique complet correspondant. Est-ce une machine de Moore ou de Mealy ?
6. Que contient la mémoire de la machine : A_{n-1}, B_{n-1} ou bien A_n et B_n ?
7. Où trouve-t-on A_n et B_n sur le schéma précédent ?
8. Exprimer P et Q en fonctions des variables d'état et des entrées.
9. Faire le schéma du dispositif (on pourra avantageusement utiliser des portes ou exclusif).
10. Pour améliorer la fiabilité du système, on veut éliminer les fluctuations des transitions des signaux A et B dues aux vibrations et aux défauts de construction. Le compteur suivant ce décodeur est synchrone et il est donc préférable qu'il ne soit soumis qu'à des signaux synchronisés sur l'horloge : donc, les signaux bruts A et B doivent être synchronisés avec l'horloge H. Faire le nouveau schéma du dispositif.

TD Annexe

Extrait de la notice du registre 74xx194

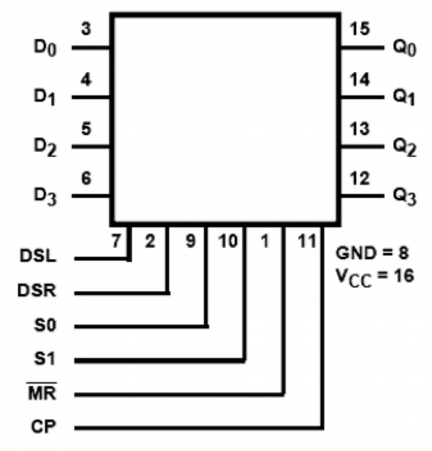
The 74HC194 and CD74HCT194 are 4-bit shift registers with Asynchronous Master Reset (MR). In the parallel mode (S0 and S1 are high), data is loaded into the associated flip-flop and appears at the output after the positive transition of the clock input (CP). During parallel loading, serial data flow is inhibited. Shift left and shift right are accomplished synchronously on the positive clock edge with serial data entered at the shift left (DSL) serial input fot the shift right mode, and at the shift right (DSR) serial input for the shift left mode. Clearing the register is accomplished by a Low applied to the Master Reset (MR) pin.

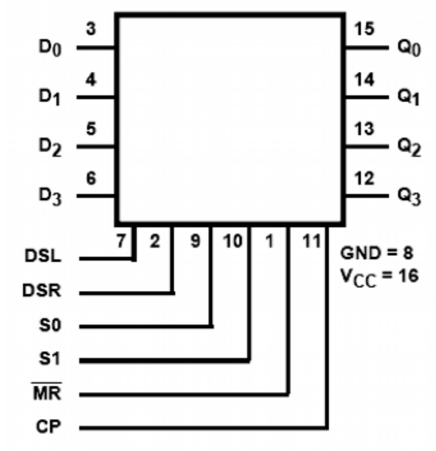


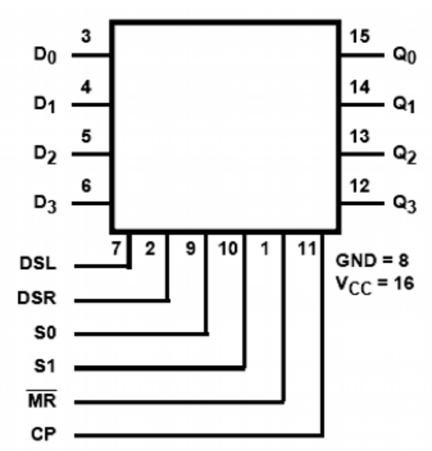
TRUTH TABLE

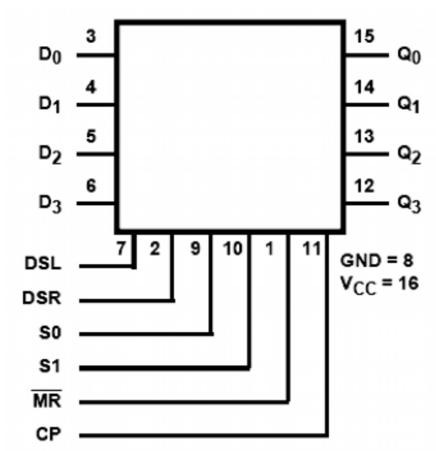
OPERATING MODE	INPUTS							OUTPUT			
	CP	$\overline{MR}$	S1	S0	DSR	DSL	D_n	Q_0	Q_1	Q_2	Q_3
Reset (Clear)	X	L	X	X	X	X	X	L	L	L	L
Hold (Do Nothing)	X	H	l (Note 1)	l (Note 1)	X	X	X	q_0	q_1	q_2	q_3
Shift Left	↑	H	h	l (Note 1)	X	l	X	q_1	q_2	q_3	L
	↑	H	h	l (Note 1)	X	h	X	q_1	q_2	q_3	H
Shift Right	↑	H	l (Note 1)	h	l	X	X	L	q_0	q_1	q_2
	↑	H	l (Note 1)	h	h	X	X	H	q_0	q_1	q_2
Parallel Load	↑	H	h	h	X	X	d_n	d_0	d_1	q_2	d_3

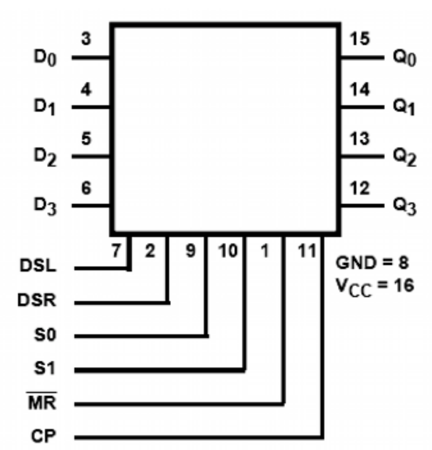
H = High Voltage Level,
h = High Voltage Level One Set-up Time Prior To The Low to High Clock Transition,
L = Low Voltage Level,
l = Low Voltage Level One Set-up Time Prior to the Low to High Clock Transition,
d_n (q_n) = Lower Case Letters Indicate the State of the Referenced Input (or output) One Set-up Time Prior to the Low To High Clock Transition,
X = Don't Care,
↑ = Transition from Low to High Level

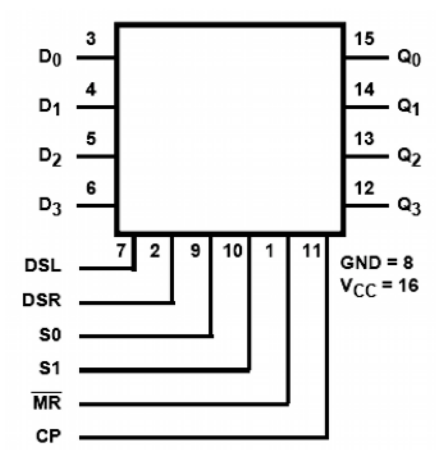

D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16


D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16


D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16


D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16


D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16


D_0
D_1
D_2
D_3
3
4
5
6
15
14
13
12
Q_0
Q_1
Q_2
Q_3
7 2 9 10 1 11
DSL
DSR
S0
S1
$\overline{MR}$
CP
GND = 8
V_{CC} = 16