# Exercices d'électronique numérique.

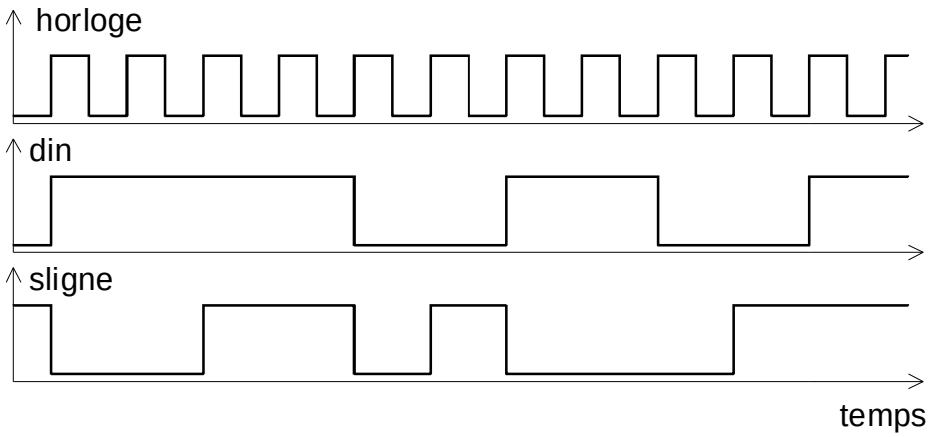
# Synthèse.

## 1. Exercice de synthèse : codeur CMI

Dans les transmissions numériques par par infra-rouge, télécommandes par exemple, on utilise souvent un code d'émission, dit code ligne tel que chaque bit émis est transmis sur *deux* périodes de l'horloge. On se propose ici de réaliser un codeur qui transforme des données binaires d'entrée, `din`, qui arrivent à raison d'un bit toutes les *deux* périodes d'horloge, en un code ligne, `sligne`, qui est construit de la façon suivante :

```
si din = '0' : sligne = '0' pendant une période d'horloge
   puis sligne = '1' pendant une période d'horloge.
si din = '1' : sligne = '0' pendant deux périodes d'horloge
   ou sligne = '1' pendant deux périodes d'horloge,
   en alternance.
```

L'alternance signifie que le niveau correspondant à un '1' logique pour din change d'une fois à l'autre, que les '1' successifs soient ou non séparés par des '0'. Le chronogramme ci-dessous donne un exemple de transmission :




Le codeur qui transforme `din` en `sligne` reçoit en entrée l'horloge et `din`. Il fournit en sortie `sligne`, retardée d'une période d'horloge, car le plus simple est de concevoir une machine de MOORE.

a A combien d'états internes correspond l'émission d'un bit ?
b Pourquoi les états correspondant à l'émission des codes pour des valeurs din = '1' successives ne peuvent ils pas être toujours les mêmes ? (évident)

c Pourquoi les états correspondant à l'émission des codes pour din = '0' ne peuvent ils pas être toujours les mêmes ? (question plus difficile)
d Déduire de l'analyse précédente le nombre d'états que doit posséder le codeur.
e Proposer une ébauche de diagramme de transitions. On nommera les états par des noms, par exemple pzero0 et pzero1 pour l'émission d'un zéro, dans l'un des cas analysée au point c. A ce niveau on représentera les transitions importantes, mais pas forcément toutes les transitions possibles.
f Le codeur peut, en début d'émission, ne pas être synchronisé correctement. Compléter le diagramme précédent pour garantir qu'il se synchronise aussi vite qu'il peut le faire.
g Choisir un codage intelligent pour les états.
h Proposer une solution VHDL au problème.
i Proposer une solution pour le décodeur.

## 2. Synthèse : codeur AMI (texte de TP VHDL)

## 3. Analyse : comparaison de deux réponses à un même problème.

Pour réaliser un codeur AMI (la connaissance de ce code n'est pas nécessaire à la compréhension du sujet), deux concepteurs différents proposent deux solutions, toutes deux justes, concrétisées par deux architectures (amidec et amidir) décrivant la même entité. Les résultats de simulation fonctionnelle des deux solutions sont identiques (voir figures en annexe).

Les programmes sources sont donnés en annexe.

Chacune de ces deux solutions est synthétisée et implémentée dans un circuit programmable AMD 16V8H-25 ($t_{PD}$ = 25 ns, $t_{CO}$ = 12 ns, $F_{maxint}$ = 40 MHz).

a Pour chaque solution indiquer la structure (pas les équations détaillées) du circuit généré. Préciser soigneusement les natures, combinatoires ou séquentielles, des différents blocs fonctionnels. Déduire de chaque programme la dimension du registre d'état correspondant. Construire les diagrammes de transitions associés.
b On teste les circuits réalisée avec une horloge à 40 Mhz.

Les résultats des deux tests sont fort différents, comme l'attestent les chronogrammes fournis en annexe. Interprétez quantitativement ces chronogrammes en vous appuyant sur les documents constructeur. Pour faciliter l'analyse on a placé des curseurs à des instants intéressants, et demandé l'affichage de l'écart temporel entre les curseurs.

c Conclure.

**Annexe : programmes**

entité commune :

```
entity amicod is
      port(       hor, din : in bit ;
            plusout, moinsout : out bit );
end amicod ;
```

architecture « amidec » :

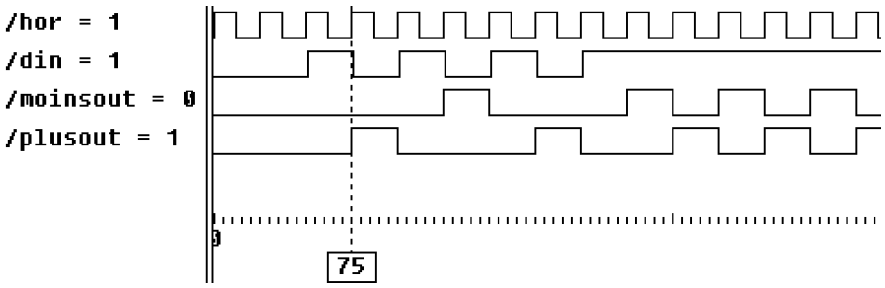
```
architecture amidec of amicod is
type ami is (mzero,pzero,moins,plus) ;
signal etat : ami ;
begin
plusout <= '1' when etat = plus else '0' ;
moinsout <= '1' when etat = moins else '0' ;
encode : process
begin
   wait until hor = '1' ;
```

```
      case etat is
         when mzero =>    if din = '1' then etat <= plus ;
               end if ;
         when pzero =>    if din = '1' then etat <= moins ;
               end if ;
         when moins =>    if din = '1' then etat <= plus ;
               else etat <= mzero ;
               end if ;
         when plus =>    if din = '1' then etat <= moins ;
               else etat <= pzero ;
               end if ;
         when others =>   etat <= mzero ;
      end case ;
   end process encode ;
   end amidec ;
```

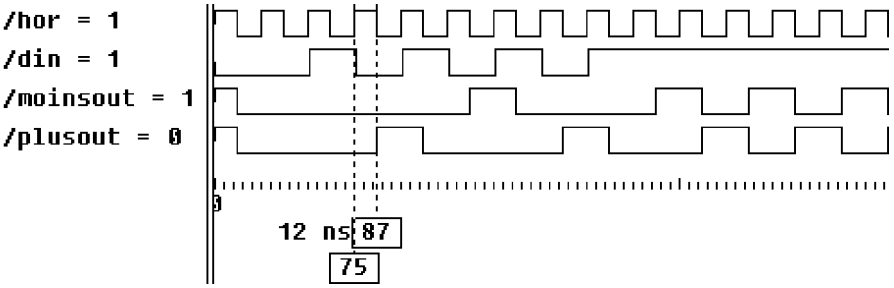
architecture « amidir » :

```
   architecture amidir of amicod is
   subtype ami is bit_vector(2 downto 0) ;
   constant mzero : ami := "000" ;
   constant pzero : ami := "001" ;
   constant moins : ami := "010" ;
   constant plus  : ami := "100" ;
   signal etat : ami ;
   begin
   plusout <= etat(2) ;
   moinsout <= etat(1) ;
   encode : process
   begin
      wait until hor = '1' ;
      case etat is
         when mzero =>    if din = '1' then etat <= plus ;
               end if ;
         when pzero =>    if din = '1' then etat <= moins ;
               end if ;
         when moins =>    if din = '1' then etat <= plus ;
               else etat <= mzero ;
               end if ;
         when plus =>    if din = '1' then etat <= moins ;
               else etat <= pzero ;
               end if ;
         when others =>   etat <= mzero ;
      end case ;
   end process encode ;
   end amidir ;
```

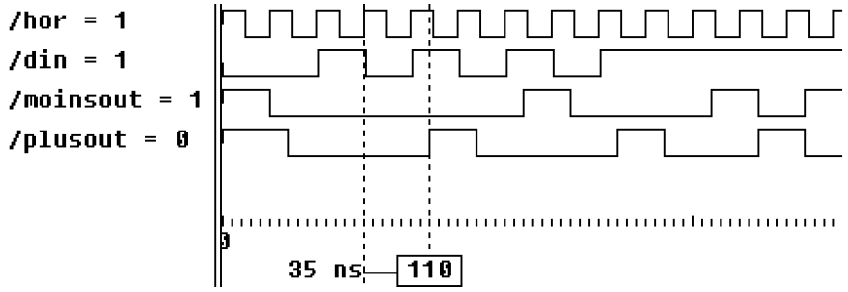
**amidec ou amidir : simulation fonctionnelle du programme source**




**amidir : fonctionnement du circuit**



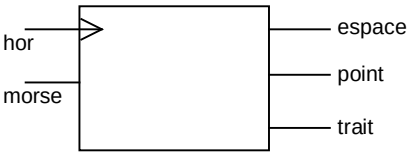

**amidec : fonctionnement du circuit**

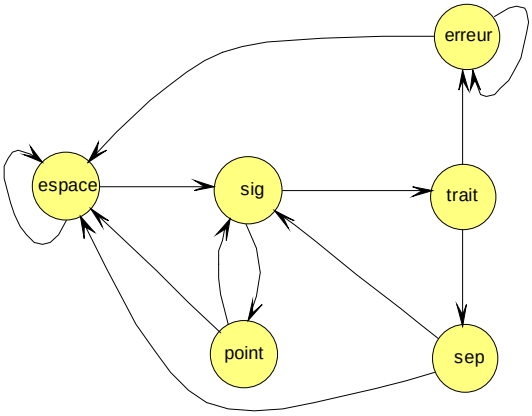


## 4. Décodeur de signaux Morse

L'alphabet morse est construit à partir de deux motifs : les points et les traits[1]. Dans une transmission en morse on réalise le point par un '1' logique qui dure une période d'horloge, le trait par un '1' logique qui dure deux périodes d'horloge. Entre deux motifs d'un même caractère le '0' dure une période d'horloge, entre deux caractères le zéro dure au moins deux périodes d'horloge.

On veut réaliser un automate qui reçoit en entrée le code morse (des impulsions qui arrivent en série) et qui fournit en sortie trois signaux : un indicateur d'espace entre lettres, un indicateur de point, un indicateur de trait. Ces indicateurs, actifs à '1', durent une période d'horloge. Si le signal d'entrée est à '1' pendant plus de deux périodes d'horloge l'automate se met en position d'erreur, il n'en sort que quand le signal d'entrée revient à '0'.




Une ébauche de diagramme de transitions est fourni ci-dessus.

1 Expliquez à quoi correspondent les états du diagramme.
2 Compléter le diagramme par des conditions sur les transitions.
3 Dans ce diagramme seuls deux états peuvent durer plus d'une période d'horloge, lesquels ?
4 Combien de bascules sont-elles nécessaires au minimum ?
5 Si on choisit de générer chaque signal de sortie par une sortie de bascule combien faut-il de bascules ?
6 Proposer un programme VHDL qui réalise l'automate.

Question subsidiaire :

7 Proposer l'automate qui décode l'alphabet…

[1] Pour les amateurs :

| | | | | |
|---|---|---|---|---|
| A | . − | | N | − . |
| B | − . . . | | O | − − − |
| C | − . − . | | P | . − − . |
| D | − . . | | Q | − − . − |
| E | . | | R | . − . |
| F | . . − . | | S | . . . |
| G | − − . | | T | − |
| H | . . . . | | U | . . − |
| I | . . | | V | . . . − |
| J | . − − − | | W | . − − |
| K | − . − | | X | − . . − |
| L | . − . . | | Y | − . − − |
| M | − − | | Z | − − . . |

## 5. Codeur HDB3

Dans les liaisons téléphoniques numériques à haut débit (2, 8 et 34 Mbits/s) on utilise en Europe un code à trois niveaux dérivé du code AMI (voir poly de travaux pratiques) :

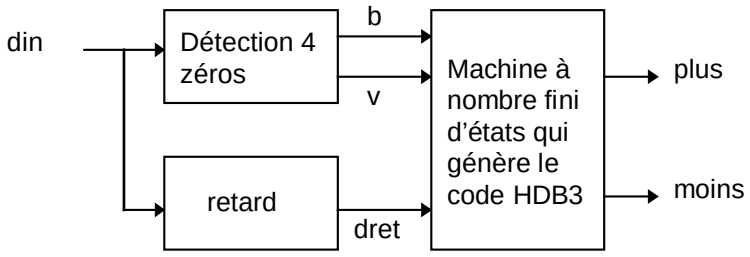
- Les bits d'information égaux à '1' sont transmis par des impulsions alternativement positives ou négatives, de façon à maintenir une valeur moyenne nulle pour le signal véhiculé sur la ligne. Les bits égaux à '0' correspondent à une différence de potentiel nulle sur la ligne (« absence » d'impulsion).
- L'horloge de réception utilise les transitions du signal reçu pour maintenir une bonne synchronisation ; des longues séquences de '0' risquent donc de créer des glissements entre l'horloge d'émission et l'horloge de réception. Pour remédier à ce problème, toutes les séquences de quatre '0' successifs sont codées par des motifs (suites d'impulsions et de niveaux nuls) qui, pour être différenciés des motifs « normaux », ne respectent pas les règles d'alternance. On parle de *violation*.
- Pour assurer le maintien de la valeur moyenne nulle, deux violations successives doivent être de polarités opposées. Si cette alternance ne correspond pas à la situation « naturelle », c'est un problème de parité du nombre des '1' qui séparent deux groupes successifs de quatre '0', on introduit dans le motif un élément de *bourrage* avant l'élément de « viol ».

Pour résumer, toute séquence de quatre zéros consécutifs génère dans le code de sortie le motif générique "b00v", où b est une éventuelle impulsion de bourrage (qui respecte la règle d'alternance) et v une impulsion (toujours présente) qui ne respecte pas la dite règle.

Dans l'exemple ci-dessous on représente les signaux ternaires émis par +, 0 et – :

| **données** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **code émis** | **0** | **0** | **+** | **0** | **0** | **0** | **+** | **–** | **+** | **–** | **0** | **0** | **–** | **+** | **–** | **+** | **0** | **0** | **+** | **–** | **0** | **0** | **0** | **–** |
| **bourrage viol** | | | | | | | **v** | | | **b** | | | **v** | | | **b** | | | **v** | | | | | **v** |

La structure du codeur à réaliser est la suivante :



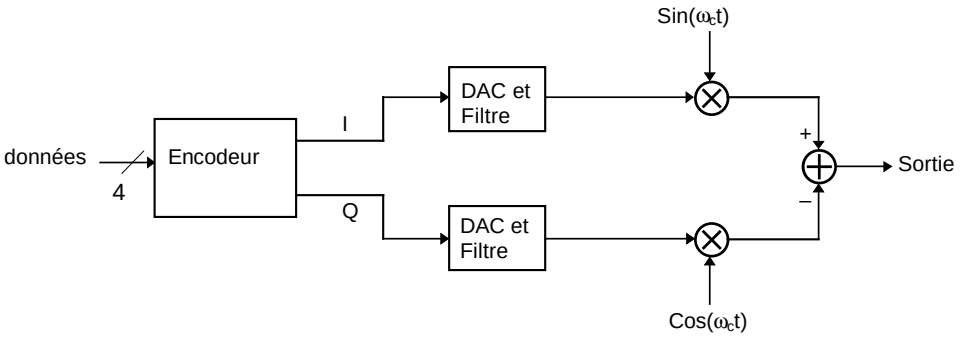

Sur ce schéma de principe on n'a pas représenté l'horloge d'émission qui synchronise l'arrivée des données et l'ensemble du système. Les signaux b et v indiquent ici les positions que doivent avoir les éventuels motifs de bourrage et de viol. Les signaux binaires de sortie plus et moins servent à générer des impulsions de polarités correspondantes.
Le bloc détecteur de quatre zéros n'ayant pas la faculté de dire l'avenir, il est nécessaire de retarder d'un nombre ad-hoc de périodes d'horloge le signal d'entrée du codeur.

Une ébauche d'algorithme décrivant le codeur proprement dit pourrait être quelque chose du genre :

```
Si le dernier viol est positif
          Si la dernière impulsion de sortie est positive
                    générer un bourrage négatif,
                    générer un viol négatif.
          Autrement
                    ne pas générer de bourrage,
                    générer un viol négatif.
Autrement
          Si la dernière impulsion de sortie est négative
                    générer un bourrage positif,
                    générer un viol positif.
          Autrement
                    ne pas générer de bourrage,
                    générer un viol positif.
```

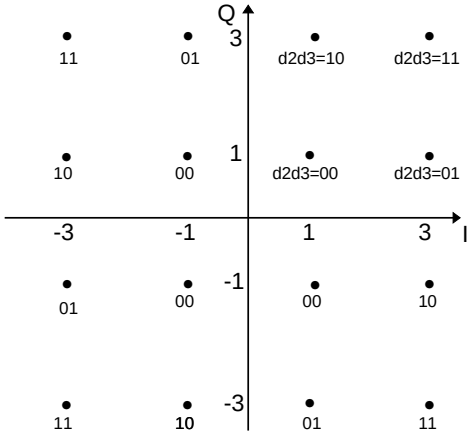
## 6. Codeur pour modulateur QAM différentiel

Dans nombre de systèmes de transmissions numériques, l'objectif est de transmettre le plus haut débit binaire possible dans un canal de bande passante donnée (et compte tenu d'un rapport signal/bruit non infini).
Une méthode classique consiste à moduler en amplitude et en phase (modulation vectorielle) une porteuse. Le « vecteur de Fresnel » associé à la porteuse peut avoir un nombre fini de valeurs, les points représentatifs de l'extrémité de ce vecteur, dans le plan complexe, constituent une constellation qui caractérise la modulation. Les informations binaires d'entrée provoquent des changements d'état dans la constellation, changements qui obéissent à une règle de codage.
Le schéma de principe typique d'un tel système est représenté ci-dessous.




L'exercice proposé (inspiré des modems téléphoniques) consiste à réaliser un codeur différentiel qui fixe des trajectoires dans une constellation à 16 états en calculant les signaux I (in phase) et Q (quadrature), codés sur deux bits ($I_2I_1$ et $Q_2Q_1$), en fonction des données d'entrée qui sont regroupées par paquets de quatre bits ($d_3d_2d_1d_0$).

La constellation du modulateur est décrite ci-dessous :




Les points sont répartis dans quatre quadrants ; les valeurs de d2 et d3 déterminent le point dans le quadrant concerné, celles de d0 et d1 déterminent les changements de quadrant entre un code et le suivant (codage différentiel de la phase) :

| d0 d1 | changement de phase |
|---|---|
| 0 0 | + 90° |
| 0 1 | 0 |
| 1 1 | +270° |
| 1 0 | +180° |

Les valeurs de I et Q sont des entiers signés, théoriquement sur trois bits, mais les valeurs étant toujours impaires, le poids faible est une constante câblée de façon fixe à l'entrée des convertisseurs numériques analogiques.

Le travail consiste à réaliser le codeur, piloté par l'horloge qui cadence l'arrivée (par paquets de 4 bits) des données d'entrée.

## Décodeur pour modulateur QAM différentiel

Même problème que le précédent, « à l'envers ». Les données d'entrée sont les valeurs de I et Q, celles de sortie sont les valeurs de d3d2d1d0.