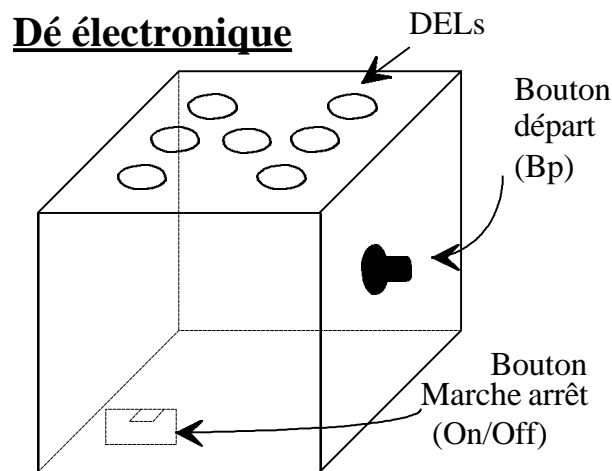


# Utilisation de PALs séquentiels

## DE ÉLECTRONIQUE

### I) Présentation du Dé électronique:

Le projet proposé (Dé électronique) remplace le dé traditionnel dans de nombreux jeu. Il se présente sous la forme d'un petit cube dans lequel sont insérées la carte électronique et la (ou les) pile (s). Nous nous limiterons ici à une version simplifiée, comportant un bouton de mise sous tension (sous le dé). Celui-ci permet de l'alimenter ou de le mettre hors tension. Un bouton poussoir sur une face (Bp = départ) permet de relancer la génération d'un nouveau chiffre. Des diodes électroluminescentes (DELs) assurent la visualisation de manière identique à un dé habituel sans que celui ci ne soit lancé.

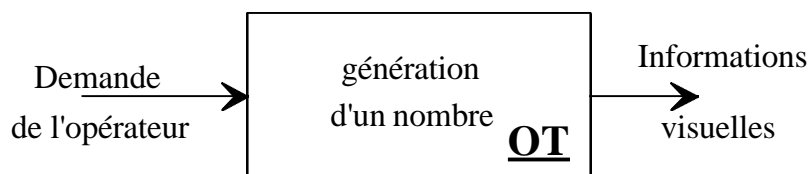


### II) Analyse fonctionnelle sommaire de l'O.T.

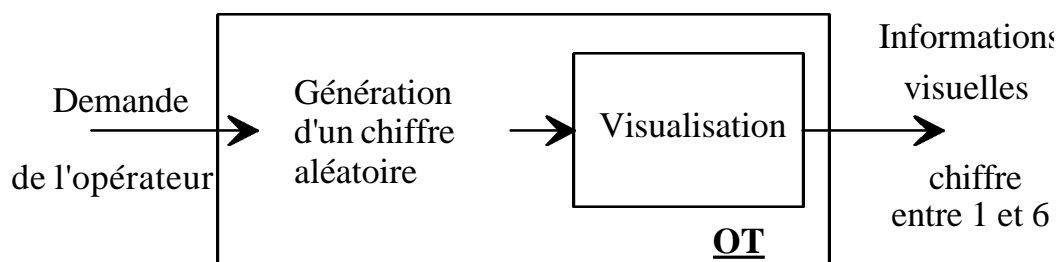
#### 1) Niveau 1

**Fonction globale:** Affichage d'un nombre, à chaque demande de l'opérateur.

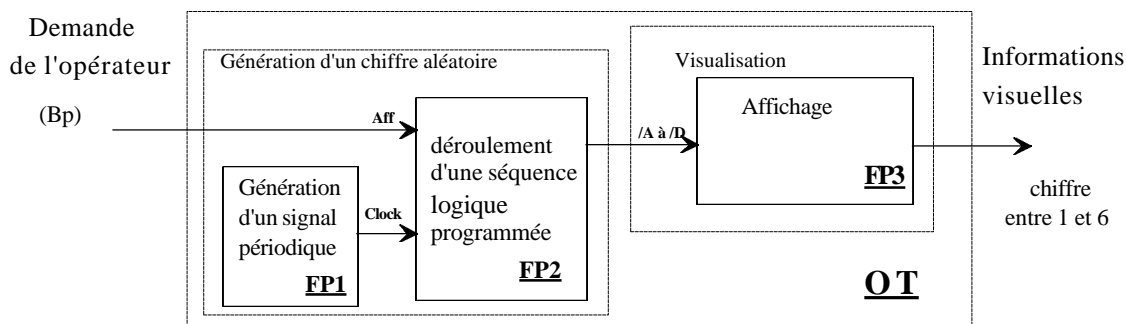
**Autres O.T. de même fonction globale:** (calculatrice, loto, roulette...).



#### 2) Niveau 2.



### 3) Schéma fonctionnel de 1<sup>er</sup> degré.



### III) Analyse structurelle et description du fonctionnement de l'O.T.

(Voir le schéma structurel du Dé électronique).

La fonction FP3 est réalisée à l'aide des diodes électroluminescentes (LED) D1 à D7. Une LED (A) est allumée lorsque la sortie /A est à 0, donc lorsque  $A=1$ . Les fonctions FP1 et FP2 seront réalisées par un seul circuit intégré en logique programmable PAL16R4, associé à quelques composants discrets (résistances et condensateurs).

La fonction FP1 (Génération d'un signal périodique = Horloge), fait appel à une structure classique d'astable. Les 2 portes logiques inverseuses nécessaires seront réalisées par le PAL. On appellera dans les équations E1, E2, S1, S2 respectivement les broches 3, 4, 12, 13 du PAL.

La fonction FP2 correspond à un bloc en logique séquentielle où les 4 sorties /A, /B, /C, /D correspondent aux 4 sorties verrouillées du PAL16R4 (broches 14, 15, 16, 17).

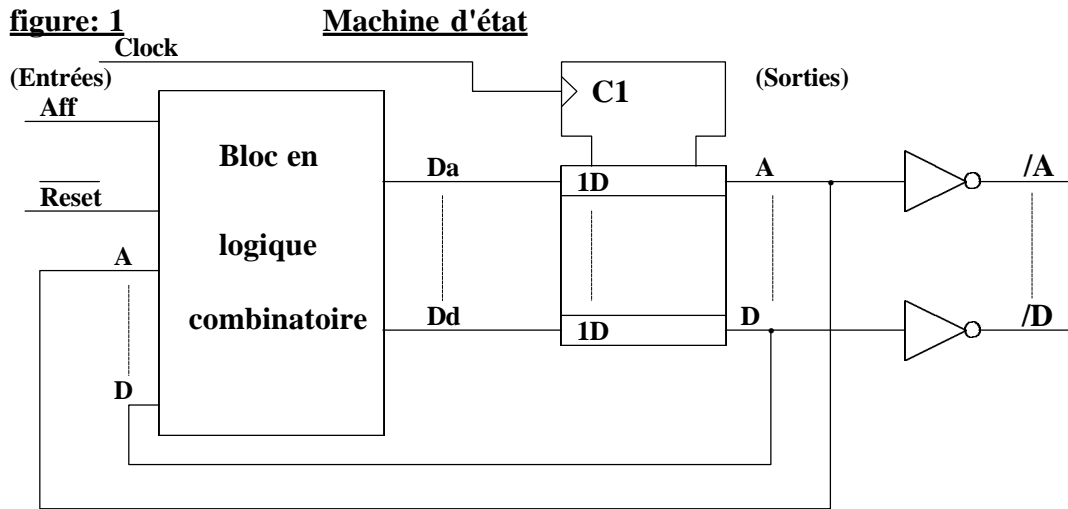
L'analyse de la structure interne de ce PAL montre la nécessité d'autoriser les sorties par la broche  $\overline{\text{Enable}}$  (broche 11 active à l'état bas), et de relier la broche Clock (broche 1) à l'oscillateur (broche 12 ou 13) de manière externe (pas de connexion interne possible).

2 broches de commande ont été ajoutées:

- $\overline{\text{Reset}}$  (broche 2) pour assurer l'initialisation correcte des bascules D à la mise sous tension à l'aide d'un circuit RC d'initialisation. A la mise sous tension les sorties des bascules A à D seront initialisées à 0 (les sorties du PAL seront donc à 1).
- Aff pour transmettre la demande de l'opérateur d'obtenir un autre chiffre. Lorsque Aff est à l'état bas la séquence se déroule à la fréquence imposée par l'horloge (choisie à 1KHz). Lorsque le bouton poussoir Bp est relâché le condensateur C3 se charge par la résistance R3. Lorsque le seuil  $V_{\text{IH}}$  est atteint, l'information Aff est alors à l'état haut, et le déroulement de la séquence est stoppé.

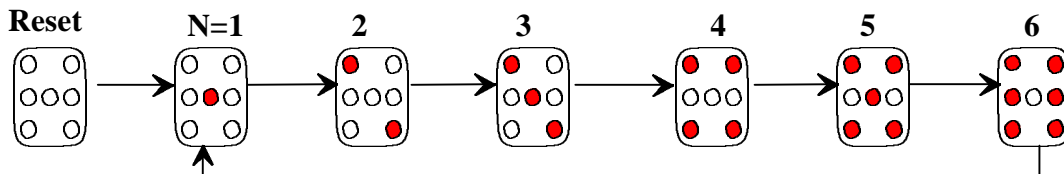
### IV) Analyse de FP2 et synthèse du PAL16R4.

La fonction FP2 est réalisée en utilisant le principe des machines d'états (Voir figure 1 page suivante).



**Dans les questions suivantes on suppose  $Aff = 0$  et  $\overline{Reset} = 1$  (déroulement séquence).**

1) A l'aide des figures du dé (ci-dessous) remplissez les colonnes A à D du tableau étape par étape, pour obtenir la séquence désirée représentée par les flèches.



N	A	B	C	D	Da	Db	Dc	Dd
Reset								
1								
2								
3								
4								
5								
6								

2) A l'aide du schéma de la machine d'état (figure 1), en déduire pour chaque étape les états logiques nécessaires sur les entrées  $Da$  à  $Dd$  des 4 bascules D, afin de passer à l'étape suivante au prochain front d'horloge (Clock).

3) Donnez alors les équations de  $Da$ ,  $Db$ ,  $Dc$ ,  $Dd$  en fonction de  $A$ ,  $B$ ,  $C$ ,  $D$ , sans chercher à les simplifier.

Ex:  $Dc = A.B.C.D + A.B.C.D + A.B.C.D$

4) Utilisez les tableaux de Karnaugh pour simplifier les équations de Da à Dd.

**Da**

		C D			
		00	01	11	10
A B	00				
	01				
	11				
	10				

**Db**

		C D			
		00	01	11	10
A B	00				
	01				
	11				
	10				

**Dc**

		C D			
		00	01	11	10
A B	00				
	01				
	11				
	10				

**Dd**

		C D			
		00	01	11	10
A B	00				
	01				
	11				
	10				

Da =

Db =

Dc =

Dd =

5) Les équations précédentes ne sont vraies que lorsque Aff = 0 et  $\overline{\text{Reset}} = 1$ .

Lorsque Aff = 0 et  $\overline{\text{Reset}} = 0$ , toutes les LEDs doivent s'éteindre. Indiquez alors les modifications apportées aux équations de Da à Dd.

6) Lorsque Aff = 1 on désire figer les sorties (verrouillage de l'affichage). Proposez alors les modifications des équations.

7) Donnez les équations de la partie horloge (S1 et S2) en fonction de E1 et E2.

8) Proposez le fichier de description du PAL.

Rappel: Le PAL16R4 à des sorties actives à l'état bas (inverseurs en sortie). L'équation correspond alors au complément de la sortie (comme avant l'inverseur).

Rem: Pour les systèmes séquentiels les équations des sorties à registre correspondent à l'état de la sortie après un front d'horloge. Ceci est signalé par les caractères { := }

Ex:  $D := \overline{D}$  veut dire: l'état de la sortie D après le front d'horloge correspondra à l'état de  $\overline{D}$  avant le front.

(D := revient écrire l'équation de l'entrée D de la bascule).

9) Comparez les équations avec le fichier De2.PDS fourni, et critiquez les. Expliquez ce qui se passe si l'on active la remise à 0 alors que l'on est en position verrouillage (Aff = 1).

10) Analysez le segment simulation et représentez les chronogrammes de Clock, /reset, Aff, A, B, C, D. Comparez avec la simulation fournie. Indiquez alors les temps de propagations représentés sur la simulation.

**Fichier DE2.PDS**

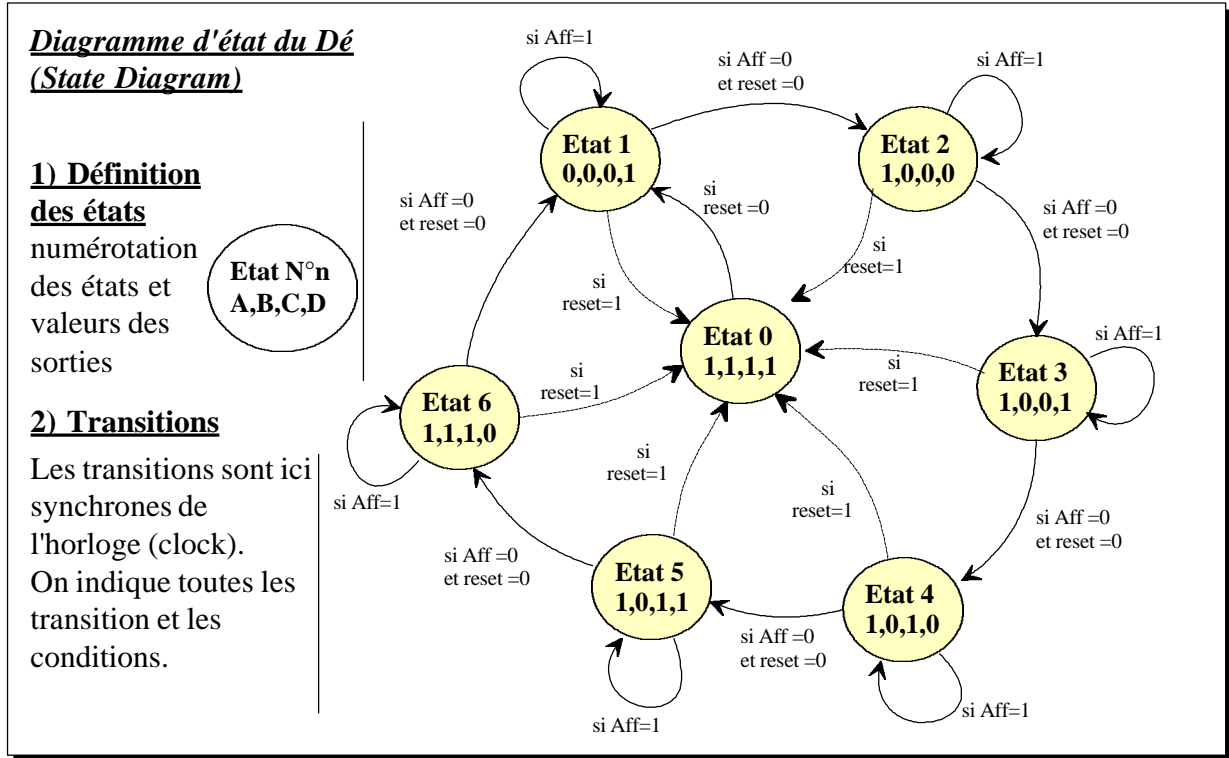
```

;PALASM Design Description
;----- Declaration Segment -----
TITLE    De electronique
PATTERN
REVISION 2.0
AUTHOR   JJ COTTET
COMPANY  Lycee Jules Ferry - VERSAILLES -
DATE     11/22/91
CHIP     _de PAL16R4
;----- PIN Declarations -----
PIN  1          CLOCK                      ;
PIN  2          /RESET                      ;
PIN  3          E1                          ;
PIN  4          E2                          ;
PIN  5          AFF                          ;
PIN  11         /ENABLE                     ;
PIN  12         S1                          ;
PIN  13         S2                          ;
PIN  14         /A                          ;
PIN  15         /B                          ;
PIN  16         /C                          ;
PIN  17         /D                          ;
;----- Boolean Equation Segment -----
EQUATIONS
/S1 = E1
/S2 = E2
A      := /A * /B * /C * D * /RESET * /AFF
        + A * /B * /RESET * /AFF
        + A * AFF
B      := A * C * D * /RESET * /AFF
        + B * AFF
C      := A * D * /RESET * /AFF
        + /B * C * /RESET * /AFF
        + C * AFF
D      := /D * /RESET * /AFF
        + D * AFF
;----- Simulation Segment -----
SIMULATION
TRACE_ON CLOCK /RESET AFF A B C D
SETF ENABLE /CLOCK RESET /AFF
FOR I := 1 TO 3 DO
    BEGIN
        CLOCKF CLOCK
    END
SETF /RESET
FOR I := 1 TO 16 DO
    BEGIN
        CLOCKF CLOCK
        IF ( I = 8 ) THEN
            BEGIN
                SETF AFF
            END
        IF ( I = 12 ) THEN
            BEGIN
                SETF /AFF
            END
    END
TRACE_OFF
;-----

```

## V) Ecriture de la machine d'état en langage ABEL et synthèse sur un GAL16V8.

La fonction FP2 (déroulement de la séquence logique programmée) est réalisée en utilisant le principe des machines d'états, et correspondant au diagramme d'état suivant.



### 1) Partie Déclaration

#### a) Déclaration des broches, des variables et des constantes.

En vous aidant du champ déclaration ci-contre, complétez la déclaration des constantes correspondant aux états en sortie, que l'on nommera Aff0 (pour l'état 0) à Aff6 (pour l'état 6).

```
clock, !oe, reset, Aff pin ;
Abar, Bbar, Cbar, Dbar pin istype 'reg' ;

sortie = [Abar, Bbar, Cbar, Dbar] ;
Aff0= ![1,1,1,1] ;
Aff1= ![0,0,0,1] ;
```

#### b) Définitions des états et des valeurs des sorties.

Il faut définir les différents états de la machine d'état (Rem: ici les états correspondent uniquement aux valeurs des sorties déjà définies Aff0 à Aff6).

```
"State value = valeur des états ...
Etat0 = Aff0; Etat1 = Aff1;
Etat2 = Aff2; Etat3 = Aff3;
```

### 2) Partie équation

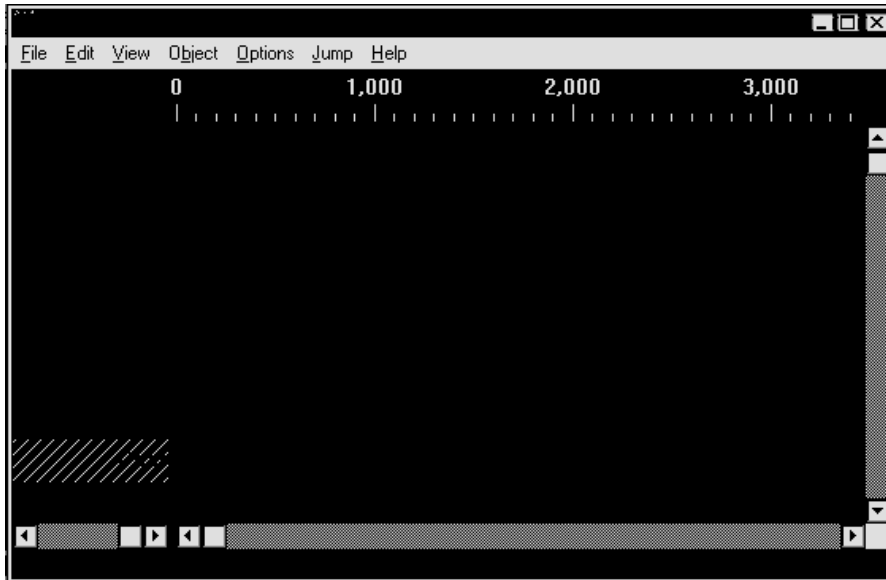
En vous aidant du champ équation ci-contre, complétez la description des transitions états.

```
equations
    sortie.clk = clock;
    sortie.oe = oe;
state_diagram sortie
    state Etat0: "si reset actif
        if (reset) then Etat0 else Etat1 ;
    state Etat1:
        if (reset) then Etat0
        else if (Aff) then Etat1 ;
        else Etat2 ;
    state Etat2:
        if (reset) then Etat0
        else if (Aff) then Etat2 ;
        else Etat3 ;
```

### **3) Partie simulation ou test.**

En vous aidant des lignes ci-contre et du chronogramme de simulation ci dessous, compléter le champ des vecteurs de test pour obtenir la simulation complète.

```
test_vectors
([clock, reset, Aff] -> sortie )
[ 0 , 1 , 0 ] -> [x,x,x,x] ;
[ 1 , 1 , 0 ] -> [x,x,x,x] ;
```



Montrez à l'aide de cette simulation, le fonctionnement ou le dysfonctionnement du dé.



```

MODULE de_16r4
TITLE 'Exercice du DE 16R4'

    clock,!oe, reset, Aff pin ;
    Abar,Bbar,Cbar,Dbar pin istype 'reg';

    sortie = [Abar,Bbar,Cbar,Dbar];
    Aff0= ![1,1,1,1];
    Aff1=![0,0,0,1];
    Aff2=![1,0,0,0];
    Aff3=![1,0,0,1];
    Aff4=![1,0,1,0];
    Aff5=![1,0,1,1];
    Aff6=![1,1,1,0];
    x=.X.;

"State value = valeur des états ...
    Etat0 = Aff0; Etat1 = Aff1;
    Etat2 = Aff2; Etat3 = Aff3;
    Etat4 = Aff4; Etat5 = Aff5;
    Etat6 = Aff6;

equations
    sortie.clk = clock;
    sortie.oe = oe;

state_diagram sortie
    state Etat0:      "si reset actif
        if (reset) then Etat0 else Etat1 ;
    state Etat1:
        if (reset) then Etat0
        else if (Aff) then Etat1 ;
        else Etat2 ;
    state Etat2:
        if (reset) then Etat0
        else if (Aff) then Etat2 ;
        else Etat3 ;
    state Etat3:
        if (reset) then Etat0
        else if (Aff) then Etat3 ;
        else Etat4 ;
    state Etat4:
        if (reset) then Etat0
        else if (Aff) then Etat4 ;
        else Etat5 ;
    state Etat5:
        if (reset) then Etat0
        else if (Aff) then Etat5 ;
        else Etat6 ;
    state Etat6:
        if (reset) then Etat0
        else if (Aff) then Etat6 ;
        else Etat1 ;

```

```

test_vectors
    ([clock, reset, Aff] -> sortie )
    [ 0 , 1 , 0 ] -> [x,x,x,x] ;
    [ 1 , 1 , 0 ] -> [x,x,x,x] ;
    [ 0 , 1 , 0 ] -> [x,x,x,x] ;
    [ 1 , 1 , 0 ] -> [x,x,x,x] ;
    [ 0 , 1 , 0 ] -> [x,x,x,x] ;
    [ 1 , 1 , 0 ] -> [x,x,x,x] ;

    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;

    [ 0 , 0 , 1 ] -> [x,x,x,x] ;
    [ 1 , 0 , 1 ] -> [x,x,x,x] ;
    [ 0 , 0 , 1 ] -> [x,x,x,x] ;
    [ 1 , 0 , 1 ] -> [x,x,x,x] ;
    [ 0 , 0 , 1 ] -> [x,x,x,x] ;
    [ 1 , 0 , 1 ] -> [x,x,x,x] ;

    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;
    [ 0 , 0 , 0 ] -> [x,x,x,x] ;
    [ 1 , 0 , 0 ] -> [x,x,x,x] ;

    [ 0 , 0 , 1 ] -> [x,x,x,x] ;
    [ 1 , 0 , 1 ] -> [x,x,x,x] ;
    [ 0 , 0 , 1 ] -> [x,x,x,x] ;
    [ 1 , 0 , 1 ] -> [x,x,x,x] ;
    [ 0 , 1 , 1 ] -> [x,x,x,x] ;
    [ 1 , 1 , 1 ] -> [x,x,x,x] ;

END

```

**Diagramme d'état du Dé**  
**(State Diagram)**

**1) Définition**

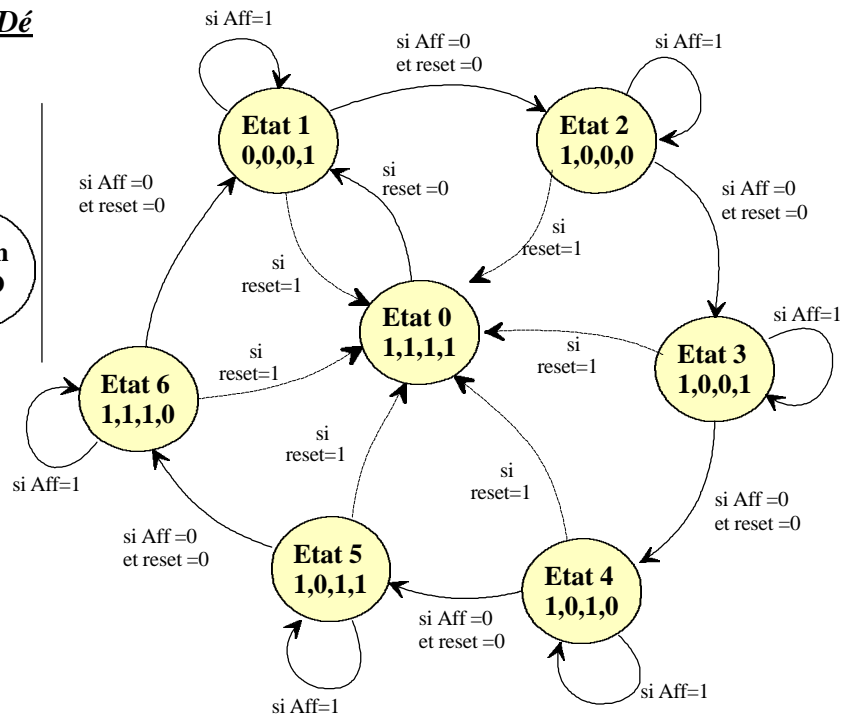
**des états**

numérotation  
des états et  
valeurs des  
sorties

Etat N°n  
A,B,C,D

**2) Transitions**

Les transitions sont ici  
synchrones de  
l'horloge (clock).  
On indique toutes les  
transition et les  
conditions.



```
clock,!oe, reset, Aff pin ;
Abar,Bbar,Cbar,Dbar pin istype 'reg';
```

```
sortie = [Abar,Bbar,Cbar,Dbar];
Aff0= ![1,1,1,1];
Aff1= ![0,0,0,1];
```

```
"State value = valeur des états ...
Etat0 = Aff0; Etat1 = Aff1;
Etat2 = Aff2; Etat3 = Aff3;
```

**equations**

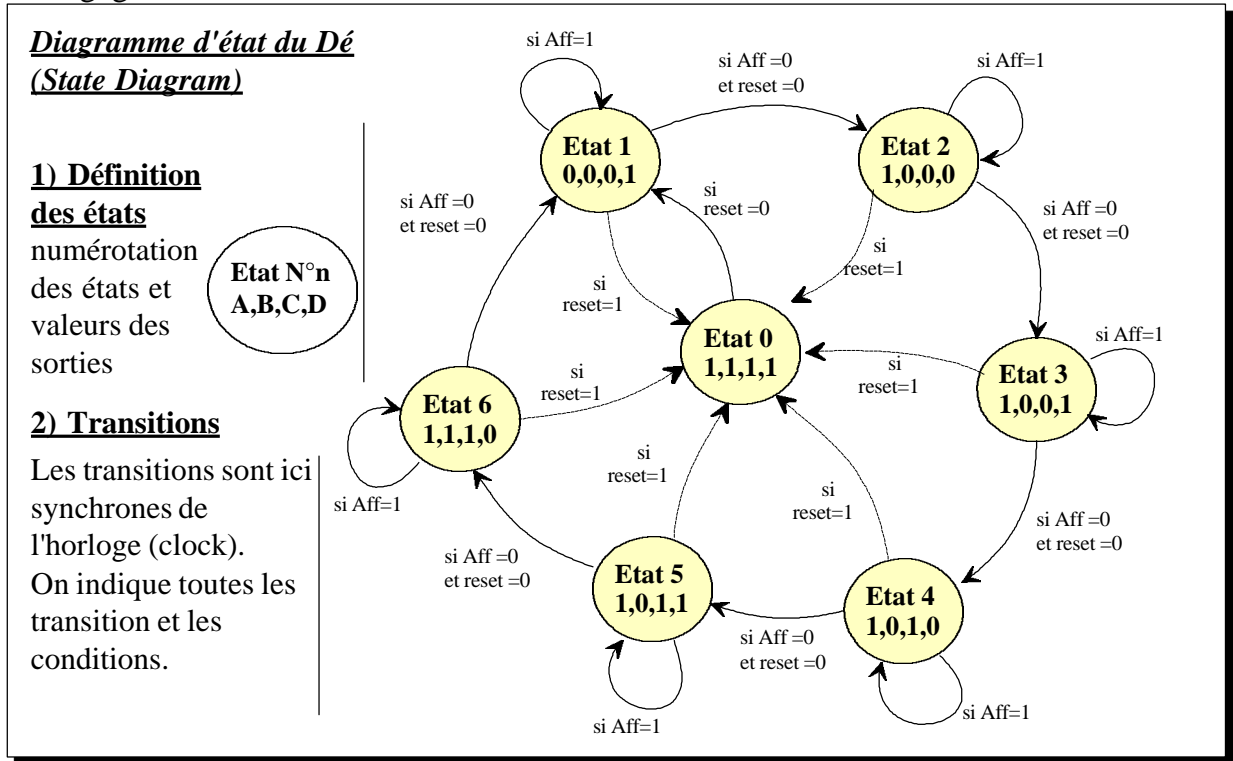
```
sortie.clk = clock;
sortie.oe = oe;
state diagram sortie
state Etat0: "si reset actif
if (reset) then Etat0 else Etat1 ;
state Etat1:
if (reset) then Etat0
else if (Aff) then Etat1 ;
else Etat2 ;
state Etat2:
if (reset) then Etat0
else if (Aff) then Etat2 ;
else Etat3 ;
```

```
test_vectors
([clock, reset, Aff] ->  sortie )
[ 0   ,   1   ,   0 ] -> [x,x,x,x] ;
[ 1   ,   1   ,   0 ] -> [x,x,x,x] ;
```



## VI) Ecriture de la machine d'état en langage VHDL et synthèse sur un GAL16V8.

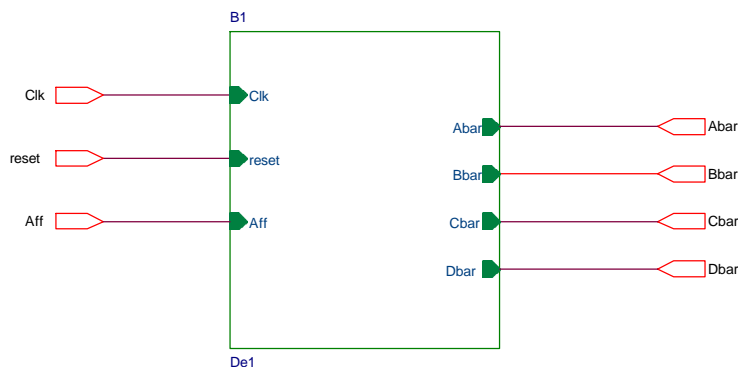
La fonction FP2 (déroulement de la séquence logique programmée) est réalisée en utilisant le principe des machines d'états, et correspondant au diagramme d'état vu précédemment en langage ABEL.



### 1) Description dans Orcad capture

#### a) Création du bloc hiérarchique.

Créez un bloc de type VHDL, placez les broches et les modules ports (PortLeft\_L et PortRight\_R).



#### b) Description en VHDL .

## PAL séquentiel (Le dé en langage VHDL sur Orcad Express)

```

-- VHDL created by OrCAD Express
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;

ENTITY Del is
  PORT (
    Clk      : IN STD_LOGIC;
    reset    : IN STD_LOGIC;
    Aff      : IN STD_LOGIC;
    Abar     : OUT STD_LOGIC;
    Bbar     : OUT STD_LOGIC;
    Cbar     : OUT STD_LOGIC;
    Dbar     : OUT STD_LOGIC);
END Del;

ARCHITECTURE behavior OF Del IS
type etats is (etat0, etat1, etat2, etat3,
etat4, etat5, etat6);
signal etat : etats ;
BEGIN
  process (Clk)
  begin
    if (Clk'event and Clk='1') then
      case etat is
        when etat0 =>
          if (reset = '1') then etat <= etat0;
          else etat <= etat1 ;
          end if ;
        when etat1 =>
          if (reset = '1') then etat <= etat0;
          else if (Aff = '1') then etat <= etat1 ;
          else etat <= etat2;
          end if ;
        when etat2 =>
          if (reset = '1') then etat <= etat0;
          else if (Aff = '1') then etat <= etat2 ;
          else etat <= etat3;
          end if ;
        when etat3 =>
          if (reset = '1') then etat <= etat0;
          else etat <= etat4;
          end if ;
        when etat4 =>
          if (reset = '1') then etat <= etat0;
          else if (Aff = '1') then etat <= etat4 ;
          else etat <= etat5;
          end if ;
        when etat5 =>
          if (reset = '1') then etat <= etat0;
          else if (Aff = '1') then etat <= etat5 ;
          else etat <= etat6;
          end if ;
        when etat6 =>
          if (reset = '1') then etat <= etat0;
          else if (Aff = '1') then etat <= etat6 ;
          else etat <= etat1;
          end if ;
        end if ;
      end case;
      etat <= etat1;
    end if;
  end process;

  Abar <= '1' when (etat = etat1) else '0';
  Bbar <= '0' when ((etat = etat0) or (etat = etat6)) else '1';
  Cbar <= '1' when ((etat = etat1) or (etat = etat2) or (etat = etat3)) else '0';
  Dbar <= '1' when ((etat = etat2) or (etat = etat4) or (etat = etat6)) else '0';
END behavior;

```

