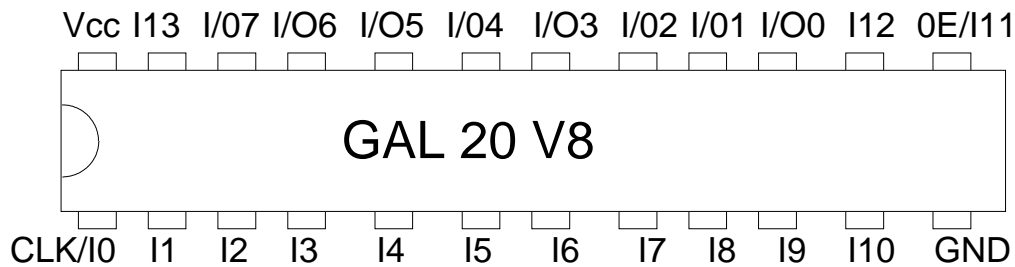


PROGRAMMATION DE PALs (7 séances)

On utilisera naturellement VHDL pour la programmation des PALs. On rappelle que pour programmer une PAL il faut un fichier JEDEC, que celui-ci peut être créé par une compilation sur une cible 20V8 ou 22V10.

I) Présentation du circuit (GAL 20V8)

Nous utiliserons des circuits en technologie CMOS appelés parfois GAL. Le circuit utilisé est un 20v8 comprenant 24 broches dont le brochage est donné ci-dessous :



Le nom 20v8 s'explique comme ceci : 20 est le nombre d'entrées possibles sur le tableau de fusibles, v signifie versatile qui veut dire que l'on a des cellules programmables et 8 est le nombre des sorties.

Le concept de cellules programmables fait l'intérêt de ces circuits. La sortie peut se faire directement, ou passer par des bascules D, être complémentée ou non. Il est donc possible de faire de la logique combinatoire ou séquentielle. Et de plus ces composants peuvent être effacés et reprogrammés.

II) Combinatoire

1°) Passage table de vérité -> équations

Modification sur le langage VHDL pour une programmation.

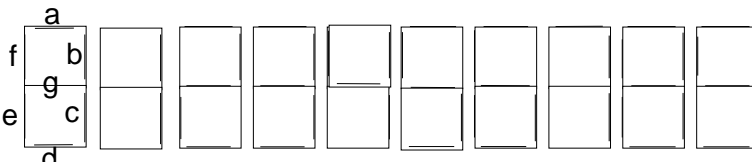
Il faut spécifier les broches d'entrées et de sorties :

```
ENTITY decod7seg IS
  PORT(D0,D1,D2,D3 : IN BIT;
        a,b,c,d,e,f,g : OUT BIT);
  ATTRIBUTE pin_numbers OF decod7seg:ENTITY IS
    "D0:2 D1:3 D2:4 D3:5 " &
    "a:15 b:16 c:17 d:18 e:19 f:20 g:21";
END decod7seg;
ou
ENTITY decod7seg IS
  PORT(D : IN BIT_VECTOR(3 DOWNTO 0);
        S : OUT BIT_VECTOR(6 DOWNTO 0));
  ATTRIBUTE pin_numbers OF decod7seg:ENTITY IS
    "D(0):2 D(1):3 D(2):4 D(3):5 " &
    "S(0):15 S(1):16 S(2):17 S(3):18 S(4):19 S(5):20 S(6):21";
END decod7seg;
```

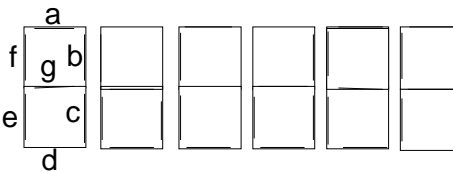


Manipulation (voir étape 9 du poly a2i11)

A) Réaliser un transcodeur BCD - 7 segments et le simplifier. Les sorties non spécifiées ci-contre sont naturellement indéterminées.



B) Réaliser maintenant un transcodeur hexadécimal 7 segments.



2°) Utilisation de l'afficheur 3 digits 1/2 (voir étape 8 pour la doc de l'afficheur)

Refaire un peu le même travail pour ces nouveaux afficheurs :

A) réalisation d'un affichage 0 à F

B) réalisation d'un affichage 0 à 15 sur 1 digit 1/2

Pourquoi ne peut-on pas utiliser une 20V8 ?

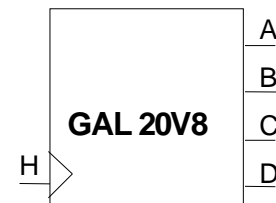
Cas du séquentiel

Avec le langage VHDL il existe plusieurs styles de programmation du séquentiel. Le plus simple est d'utiliser le style équations de récurrence.

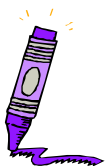
III) Compteur Gray

On désire réaliser un compteur sur 4 bits dont la sortie donne directement le code Gray (Reportez-vous au polycopié a2i11 TD 6 (p 14) pour le code Gray. Il s'agit d'un circuit qui ne comporte qu'une entrée horloge et donc 4 sorties. La déclaration des broches sera donc du type :

```
ENTITY cmptgray IS
  PORT(CLK : IN BIT;
        A,B,C,D : INOUT BIT);
  ATTRIBUTE pin_numbers OF cmptgray:ENTITY IS
    "CLK:1 A:15 B:16 C:17 D:18";
END cmptgray;
```

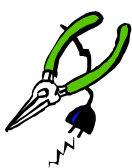


Il n'y a qu'une entrée : l'horloge, et quatre sorties.



Préparation

Écrire les équations de récurrence en fonction des sorties A, B, C et D, A étant considéré comme le poids faible. A l'aide du schéma interne de la GAL, (pages 5 et 6 du polycopié a2i13) dire ce qu'il faut faire de l'entrée OE dans le cas où l'on a des équations séquentielles.



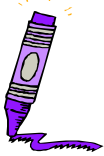
Manipulation

1°) Écrire ces équations dans le langage VHDL. Essayez votre compteur sur le simu.

2°) Reprendre le même problème avec entrée d'initialisation synchrone vers l'état 0.

V) Compteur décimal à sortie directe 7 segments

Nous désirons réaliser ici un compteur mais qui donne sa sortie directement sur un afficheur 7 segments. Ce compteur comporte donc 7 sorties (a, b, c, d, e, f et g). Un tel compteur possède un nombre d'état de $2^7=128$ mais n'en utilise en fait que 10.

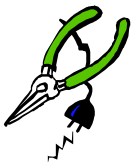


Préparation

Remplir la table de transitions ci-dessous

	Etat présent							Etat Futur						
	a	b	c	d	e	f	g	a+	b+	c+	d+	e+	f+	g+
zero	1	1	1	1	1	1	0							
un	0	1	1	0	0	0	0							
deux	1	1	0	1	1	0	1							
trois	1	1	1	1	0	0	1							
quatre	0	1	1	0	0	1	1							
cinq	1	0	1	1	0	1	1							
six	1	0	1	1	1	1	1							
sept	1	1	1	0	0	0	0							
huit	1	1	1	1	1	1	1							
neuf	1	1	1	1	0	1	1							

Écrire les équations logiques a+, b+, c+, d+, e+, f+, g+ en fonction des entrées a, b, c, d, e, f, g.
Remarque : il vous est possible d'éviter l'écriture de ces équations si vous décidez d'utiliser la structure case when



Manipulation

A) Écrire le programme VHDL. Les sorties sont imposées par la maquette.

```
ENTITY cmpt7seg IS
  PORT(CLK : IN BIT;
        a,b,c,d,e,f,g : INOUT BIT);
  ATTRIBUTE pin_numbers OF cmpt7seg:ENTITY IS
    « CLK:1 a:15 b:16 c:17 d:18 e:19 f:20 g:21 »;
END cmpt7seg;
```

B) Refaire le même travail avec une entrée d'initialisation.

Remarque :

Ce circuit existe tout fait en technologie CMOS. Il s'agit du 40110 : compteur décompteur

décimal driver 7 segments. Son schéma interne montre qu'il s'agit d'un compteur BCD suivi d'un transcodage BCD / 7 segments. Il est impossible de l'implanter comme cela dans une GAL 20v8 car il faudrait 4 sorties pour le comptage BCD + 7 pour le transcodeur = 11 sorties

VI) Programmation d'un graphe d'états

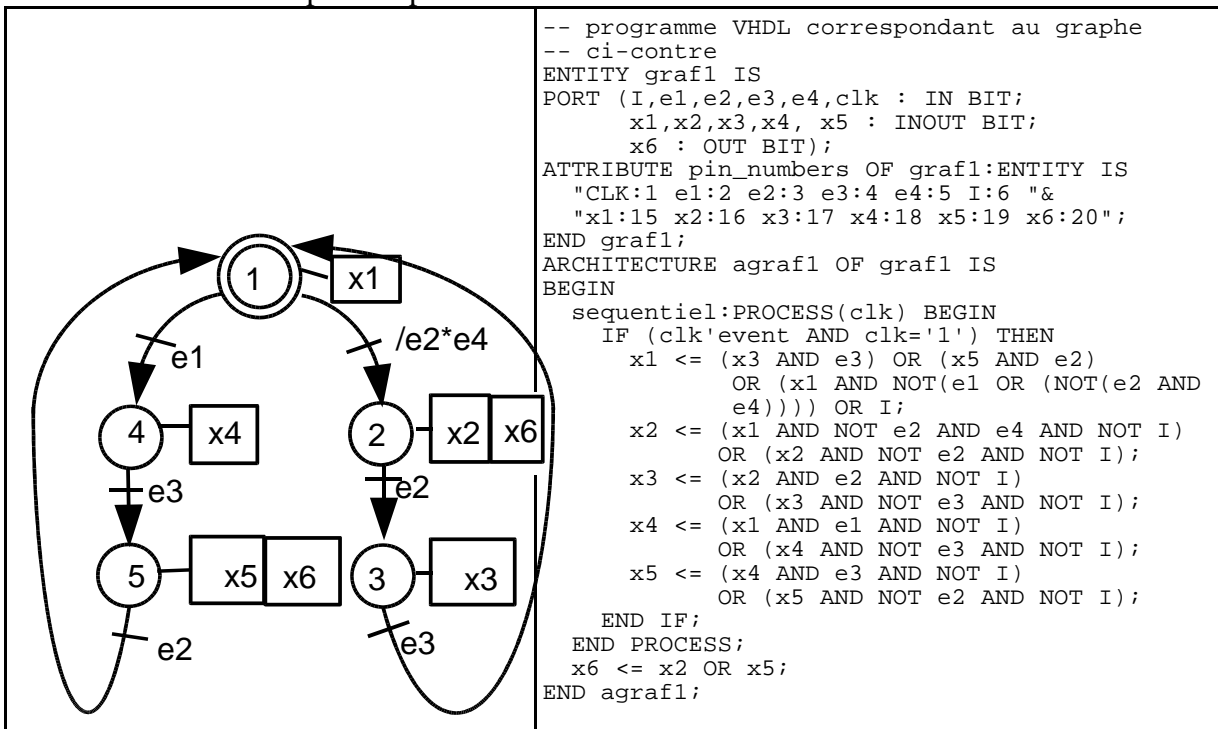
1°) graphe d'états et équations de récurrence

On rappelle qu'il est possible de transformer un graphe d'états en équations de récurrence. Du point de vue de notre langage cela implique l'utilisation de process.

Principe : pour chaque état i on écrit les conditions d'activation AC_i et de désactivation Di . Si I est l'entrée d'initialisation (permettant d'amener le graphe d'états dans son état initial), l'équation de récurrence est alors :

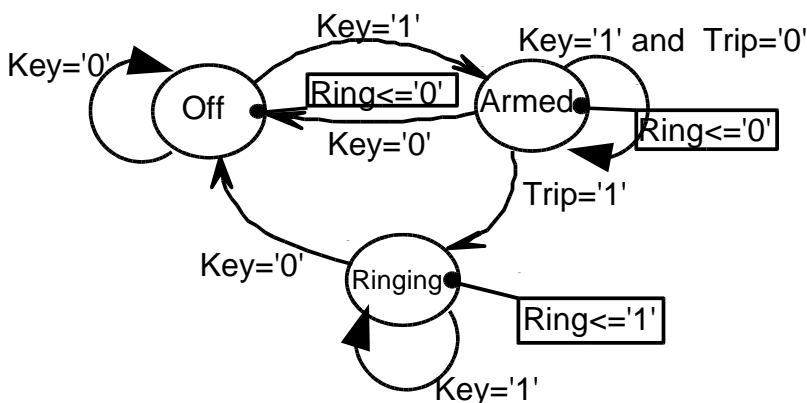
$x_i = AC_i + x_i * Di + I$ s'il s'agit d'un état initiale,
 $x_i = (AC_i + x_i * Di) * I$ s'il s'agit d'un état normal.

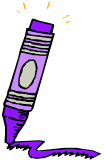
Nous donnons un exemple complet ci-dessous



2°) Matérialisation d'un graphe.

On donne le graphe de transitions du réveil TD3.





Préparation

Le transformer en graphe d'états puis donner les équations de récurrence correspondantes. Donner le schéma de câblage de la PAL.



Manipulation

Utiliser le simulateur, puis transformer en JEDEC. Essayer votre circuit sur le simulateur. Refaire ce problème en utilisant l'outil graphique Active FSM.

VI) Compteur de passages

Un ensemble de deux capteurs infrarouge perçoit le passage d'une personne et en fonction de son sens de passage incrémente ou décrémente un compteur qui décodé affiche le nombre de personnes sur deux afficheurs 7 segments.

Il nous faudra comme composants :

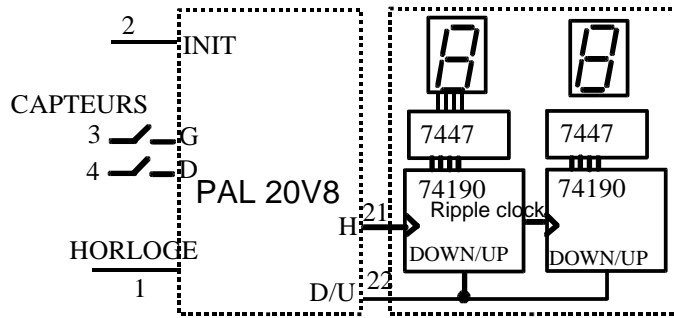
- deux décodeurs BCD -> 7 segments pour l'affichage
- deux compteurs-décompteurs décimaux (cascadés)
- une logique de contrôle pour détecter le sens de passage

Seule la logique de contrôle est réalisée avec une GALs.



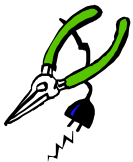
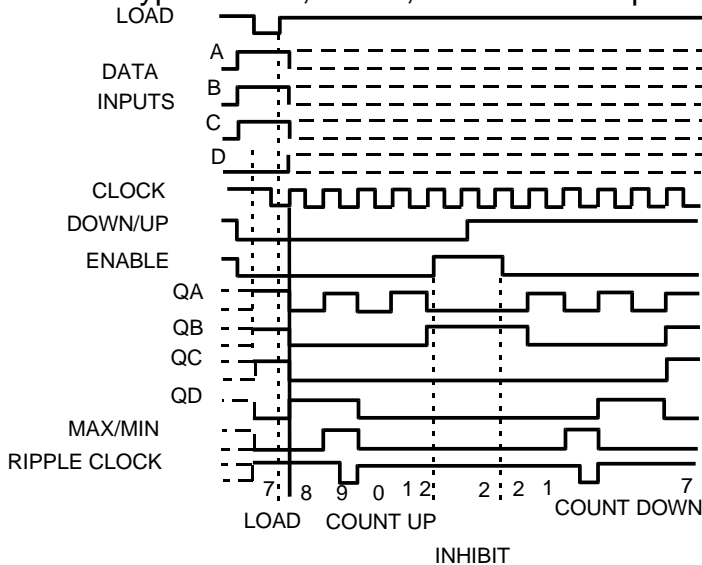
Préparation

Documentation du 74190	Travail à réaliser
<p>74190 Dual-In-Line-Package</p>	<p>Chercher à décrire la logique de contrôle par un graphe d'états.</p> <p>1°) On utilisera pour cela deux graphes de trois états. Pour les trouver essayer d'en réaliser d'abord un à trois états chargé du comptage. Puis l'autre chargé du décomptage. Ensuite on va rassembler les états initiaux. On utilisera les notations ci-dessus pour les actions (H et D/U) et les entrées (D et G).</p> <p>Écrire les équations de récurrence faisant intervenir l'entrée d'initialisation.</p> <p>2°) On cherche maintenant à réaliser ce même séquenceur mais en donnant la possibilité aux personnes de faire demi-tour entre les deux capteurs (c'est à dire de changer d'avis). Montrer qu'un graphe d'états de 9 états est nécessaire pour gérer ce problème. Écrire les équations de récurrence. Montrer qu'alors les équations ne tiennent plus dans la PAL. On utilisera donc Active-FSM pour réaliser un graphe d'états avec codage des états.</p>



LS 190 Decade Counters

Typical Load, Count, and Inhibit Sequences

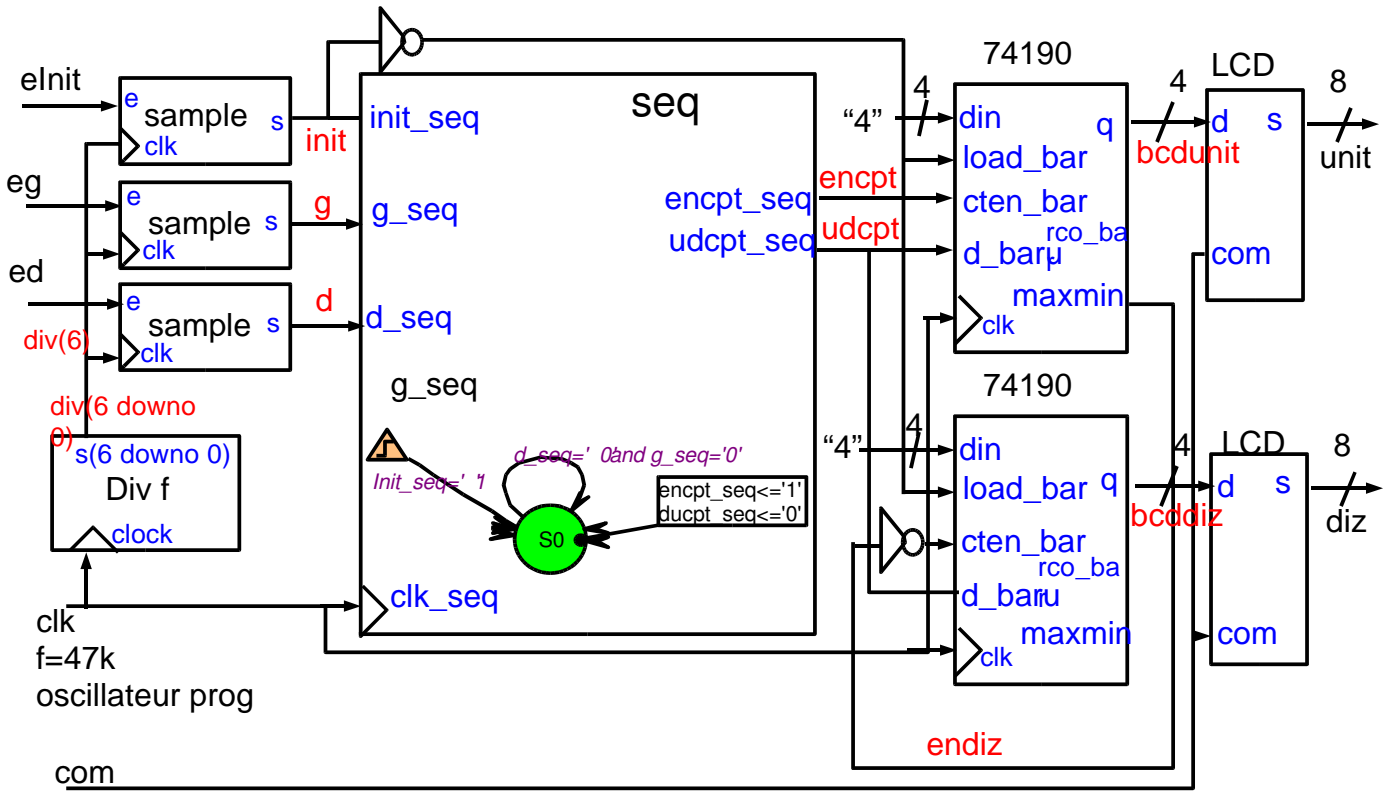


Manipulation

Programmer les deux problèmes énoncés ci-dessus. Câbler ensuite votre GAL sur le simulateur pour tester son bon fonctionnement.

Pour terminer ce TP, on vous demande de mettre le compteur de passages complet dans un FPGA Xilinx. On donne ci-après le schéma à réaliser en VHDL structurel ainsi que que le schéma de montage général.

- 1°) Ecrire le programme VHDL réalisant le 74190 complet (en utilisant des std_logic).
- 2°) Ecrire le programme VHDL structurel réalisant le fonctionnement complet.



Câblage général

