Comment programmer un PLD en ABEL sous ISIS Proteus 7 ?

Le langage ABEL permet de décrire très facilement les équations logiques combinatoires permettant de configurer des circuits logiques programmables (PLD = *Programmable Logic Device* en anglais). Dans le cas de la conception d'un logigramme complexe, la programmation en ABEL d'un PLD sera bien plus rapide que le câblage du logigramme en portes logiques, d'autant plus que la simplification des équations devient ici totalement inutile.

Installation de EZ-ABEL

Ez-Abel est un compilateur permettant de convertir un fichier source en ABEL en un fichier JEDEC exploitable par ISIS Proteus pour configurer les PLD. Télécharger le compilateur **ez-abel** sur le site **www.gecif.net** puis décompacter le fichier zip dans le répertoire :

```
C:\Program Files\Labcenter Electronics\Proteus 7 Professional\Tools\ez-abel
```

Configurer ISIS Proteus

Pour ajouter le compilateur ez-abel à la liste des générateurs de code d'ISIS Proteus il faut :

- cliquer sur **Définir outils de génération de code** dans le menu **Source** de Proteus
- cliquer sur le bouton **Nouveau** puis choisir le fichier **abel.bat** dans le répertoire `C:\Program Files\Labcenter Electronics\Proteus 7 Professional\Tools\ez-abel`
- configurer l'extension des fichiers sources : **ABL**
- configurer l'extension des fichiers objets : **JED**
- la ligne de commande étant **%1**, cliquer sur **OK**

Compiler un programme ABEL dans Proteus

Pour ajouter un nouveau fichier source ABEL directement éditable dans Proteus il faut :

- cliquez sur **Ajouter/Supprimer un fichier source** dans le menu **Source** de Proteus
- cliquer sur **Nouveau** puis choisir le fichier source (voir **exemple.abl** dans **ez-abel**)
- sélectionner l'outil de génération de code **ABEL** puis cliquer sur **OK**

Pour éditer un programme source ABEL directement dans Proteus il faut :

- cliquez sur le nom du fichier source dans le menu **Source** de Proteus
- modifier le programme source en ABEL puis l'enregistrer
- fermer l'éditeur de fichier source de Proteus

Pour compiler un programme source ABEL afin de le convertir en JEDEC dans Proteus il faut :

- cliquez sur **Tout construire** dans le menu source de Proteus

Les circuits logiques programmables disponibles dans ISIS Proteus 7.8

Proteus 7.8 dispose de 12 circuits logiques programmables (PLD) dans sa catégorie PLDs & FPGAs. Ils sont tous facilement utilisables avec ez-abel à l'exception des circuits AM16L8 et AM20V8. Dans le programme source en ABEL, le nom du **device** sera celui du circuit en remplaçant AM par P. Par exemple : **device P22V10** pour programmer le circuit **AM22V10**.

Syntaxe des opérateurs logiques en langage ABEL

En ABEL il existe 5 opérateurs logiques de base pour décrire des équations combinatoires :

Opérateur	NON	ET	OU	OU-Exclusif	OU-Exclusif-NON
Symbole	!	&	#	$	!$
Exemple	!A	A & B	A # B	A $ B	A !$ B

En ABEL les opérateurs logiques respectent les priorités habituelles. En cas de doute ne pas hésiter à ajouter dans les équations des paires de parenthèses correctement imbriquées.

Exemples de programmes en langage ABEL

Exemple 1 : réalisation d'un transcodeur décrit par ses équations logiques

```
module exemple
title 'Decodeur hexa 7 segements en ABEL - www.gecif.net - 20 fevrier 2012'

"hex_7seg est le nom du fichier JEDEC genere.
"Pour utiliser le circuit AM22V10 dans Proteus on utilise le device P22V10 en Abel :
hex_7seg      device  'P22V10';

"Declaration des 4 entrees :
        e1,e2,e3,e4        pin 1,2,3,4;

"Declaration des 7 sorties :
        a,b,c,d,e,f,g        pin 23,22,21,20,19,18,17;

"Ecriture des equations :
equations
        a = !e1&!e3#!e1&e4#e2&e3#e2&!e4#e1&e3&!e4#!e2&!e3&e4;
        b = !e1&!e3#!e3&!e4#!e1&!e2&!e4#e1&e2&!e4#e1&!e2&e4;
       c = e1&!e3#!e3&e4#e3&!e4#e1&!e2#!e2&!e4;
       d = !e2&e4#!e1&!e3&!e4#e1&e2&!e3#!e1&e2&e3#e1&!e2&e3;
       e = e2&e4#!e1&!e3#!e1&e2#e3&e4;
       f = !e1&!e2#e2&e4#!e1&e3#!e2&e3&!e4#!e3&e4;
       g = e1&e4#e2&!e3#!e3&e4#!e1&e2#!e2&e3&!e4;

end exemple
```

Exemple 2 : réalisation d'un transcodeur décrit par sa table de vérité

```
module exemple
"Cet exemple montre comment realiser un decodeur 7 segments en decrivant simplement sa table de verite
"et sans utiliser un seul operateur logique

title 'Decodeur hexa 7 segements en ABEL - www.gecif.net - 21 fevrier 2012'

"hex_7seg est le nom du fichier JEDEC genere.
"Pour utiliser le circuit AM22V10 dans Proteus on utilise le device P22V10 en Abel :
hex_7seg      device  'P22V10';

"Connexion des entrees aux bornes du circuit :
e1,e2,e3,e4           pin 1,2,3,4;

"Connexion des sorties aux bornes du circuit :
a, b, c, d, e, f, g   pin 23,22,21,20,19,18,17;

"Pour decrire une fonction logique sous forme de table de verite
"les sorties doivent etre de type combinatoire.

"Declaration des sorties en combinatoire (COM) avec le mot cle istype :
a, b, c, d, e, f, g istype 'com';

"Ecriture de la table de verite :
TRUTH_TABLE ([e4,e3,e2,e1] -> [a, b, c, d, e, f, g])
0 -> [ 1, 1, 1, 1, 1, 1, 0];
1 -> [ 0, 1, 1, 0, 0, 0, 0];
2 -> [ 1, 1, 0, 1, 1, 0, 1];
[ 0, 0, 1, 1] -> [ 1, 1, 1, 1, 0, 0, 1];
4 -> [ 0, 1, 1, 0, 0, 1, 1];
5 -> [ 1, 0, 1, 1, 0, 1, 1];
6 -> [ 1, 0, 1, 1, 1, 1, 1];
7 -> [ 1, 1, 1, 0, 0, 0, 0];
8 -> [ 1, 1, 1, 1, 1, 1, 1];
9 -> [ 1, 1, 1, 1, 0, 1, 1];
10 -> [ 0, 0, 0, 1, 0, 0, 1];
11 -> 65;
12 -> [ 1, 0, 0, 1, 0, 0, 1];
13 -> [ 0, 1, 1, 0, 1, 1, 1];
14 -> [ 1, 1, 0, 0, 0, 1, 1];
15 -> [ 0, 0, 0, 0, 0, 0, 1];

"Remarque : les entrees comme les sortie peuvent etre decrites en binaire entre crochets, ou directement
en decimal

end exemple
```

Exemple 3 : réalisation d'un séquenceur simple possédant 3 états

```
module exemple
"Cet exemple montre comment realiser un sequenceur simple en decrivant son diagramme d etat
"et sans utiliser un seul operateur logique.
"Ce sequenceur permet d alimenter successivement les 3 feux d un feu tricolore de circulation

"Realise par Jean-Christophe MICHEL le 21 fevrier 2012
"www.gecif.net

title 'sequenceur synchrone 3 bits pour feux tricolores'

"pld_seq est le nom du fichier JEDEC genere.
"Pour utiliser le circuit AM22V10 dans Proteus on utilise le device P22V10 en Abel :
pld_seq      device  'P22V10';

"les 2 entrees :
h,r pin 1,2;

"les 3 sorties sont ici de type reg_d et buffer :
q2,q1,q0 pin 23,22,21 istype 'reg_d,buffer';

"les 3 sorties forment un vecteur :
sortie=[q2,q1,q0];

"Donne un nom a chaque etat en fonction de la valeur des sorties :
Z=0;
A=1;
B=2;
C=4;

equations
"configure l'horloge sur la borne 1 :
       sortie.clk=h;
"configure le reset asynchrone sur la borne 2 :
       sortie.ar=r;

"Ecriture du diagramme d etat : apres chaque etat dans quel etat faut-il aller ?
state_diagram sortie
       state A: goto B;
       state B: goto C;
       state C: goto A;
       state Z: goto A; "apres une mise a zero on repart a l etat A

end exemple
```

Exemple 4 : réalisation d'un compteur 4 bits à cycle complet

```
module exemple
"Cet exemple montre comment realiser un compteur en ABEL
"Ce compteur compte de 0 a 15 puis recommence

"Realise par Jean-Christophe MICHEL le 21 fevrier 2012
"www.gecif.net

title 'compteur 4 bits'

"compteur est le nom du fichier JEDEC genere.
"Pour utiliser le circuit AM22V10 dans Proteus on utilise le device P22V10 en Abel :
compteur      device  'P22V10';

"les 2 entrees :
h,r pin 1,2;

"les 4 sorties sont ici de type reg_d et buffer :
q3,q2,q1,q0 pin 23,22,21,20 istype 'reg_d,buffer';

"les 4 sorties forment un vecteur :
sortie=[q3,q2,q1,q0];

equations
"configure l'horloge sur la borne 1 :
       sortie.clk=h;
"configure le reset asynchrone sur la borne 2 :
       sortie.ar=r;

"incremente le compteur seulement si l entree r n est pas a  1:
when (r==1) then sortie:=0; else sortie:=sortie+1;

end exemple
```

Exemple 5 : réalisation d'un séquenceur conditionnel (un compteur/décompteur BCD)

```
module exemple
"Cet exemple montre comment realiser un compteur/decompteur en ABEL
"Le sens de comptage est selectionne grace a une entree

"Realise par Jean-Christophe MICHEL le 21 fevrier 2012
"www.gecif.net

title 'compteur 4 bits'

"decompte est le nom du fichier JEDEC genere.
"Pour utiliser le circuit AM22V10 dans Proteus on utilise le device P22V10 en Abel :
decompte      device  'P22V10';

"les 3 entrees :
" h : l horloge
" r : le reset asy,chrone
" s : le sens de comptage (s=1 : compteur, et s=0 : decompteur)
h,r,s pin 1,2,3;

"les 4 sorties sont ici de type reg_d et buffer :
q3,q2,q1,q0 pin 23,22,21,20 istype 'reg_d,buffer';

"les 4 sorties forment un vecteur :
sortie=[q3,q2,q1,q0];

"On donne un nom a chacun des 9 etats en fonction de la valeur des sorties afin d ecrire le diagramme d
etat :
etat0=0;
etat1=1;
etat2=2;
etat3=3;
etat4=4;
etat5=5;
etat6=6;
etat7=7;
etat8=8;
etat9=9;

equations
"configure l'horloge sur la borne 1 :
       sortie.clk=h;
"configure le reset asynchrone sur la borne 2 :
       sortie.ar=r;

"Ecriture du diagramme d etat conditionnel : si s=1 on compte, et si s=0 on decompte
state_diagram sortie
       state etat0: if (s==1) then goto etat1; else etat9;
       state etat1: if (s==1) then goto etat2; else etat0;
       state etat2: if (s==1) then goto etat3; else etat1;
       state etat3: if (s==1) then goto etat4; else etat2;
       state etat4: if (s==1) then goto etat5; else etat3;
       state etat5: if (s==1) then goto etat6; else etat4;
       state etat6: if (s==1) then goto etat7; else etat5;
       state etat7: if (s==1) then goto etat8; else etat6;
       state etat8: if (s==1) then goto etat9; else etat7;
       state etat9: if (s==1) then goto etat0; else etat8;

end exemple
```

Précisions concernant l'installation et l'utilisation du compilateur EZ-ABEL :

Le programme source en ABEL doit être compilé avec ez-abel afin d'obtenir le fichier JEDEC. Le fichier JEDEC (possédant l'extension .JED) doit être indiqué dans les propriétés des PLD dans Proteus. Ez-abel se télécharge sur le site **www.gecif.net** qui propose aussi un grand nombre d'exemples d'utilisation d'ABEL.

Ez-abel étant un "vieux" compilateur des années 1990 fonctionnant en ligne de commande sous MS-DOS, son utilisation demande quelques précautions :

- Le fichier Proteus et le fichier source ABEL doivent être enregistrés tous les deux dans le même répertoire (mais possédant un chemin absolu pas trop long !)
- Le plus simple est de décompacter le fichier **ez-abel.zip** (à télécharge sur le site **www.gecif.net**) dans le répertoire ez-abel à la racine du disque dur et de travailler dans ce répertoire **c:\ez-abel**
- Le nom du fichier source .ABL doit être court (nom MS-DOS avec 8 caractères max)
- Le nom du fichier Jedec généré .JED doit être court (nom MS-DOS avec 8 caractères max)
- Pour convertir le fichier toto.abl en fichier JEDEC il faut utiliser la commande **abel toto.abl** (en ligne de commande ou avec le générateur de code de Proteus) faisant appel au fichier batch **abel.bat**