# TP de programmation du CPLD MACH 4

## *Introduction*

Il y a quelques années la réalisation d'un montage en électronique numérique impliquait l'utilisation d'un nombre important de circuits logiques. Ceci avait pour conséquences un prix de revient élevé, une mise en œuvre complexe et un circuit imprimé de taille importante.

Le développement des mémoires utilisées en informatique fut à l'origine des premiers circuits logiques programmables (PLD : programmable logic device). Ce type de produit peut intégrer dans un seul circuit plusieurs fonctions logiques programmables par l'utilisateur. Sa mise en œuvre se fait facilement à l'aide d'un micro-ordinateur et d'un logiciel adapté.

L'objectif de ce TP est de prendre en main le logiciel ispDesignExpert de la société LATTICE par l'intermédiaire d'exemples de programmation du CPLD MACH 4.

## *Les portes logiques (description structurelle)*

Ce premier exemple va permettre de prendre en main les principales fonctions du logiciel ispDesignExpert.

- Créer à partir de l'explorateur de Windows un répertoire de travail portant votre nom dans le répertoire « ispTOOLS ».
- Créer un nouveau répertoire appelé « Portes logiques » dans votre répertoire de travail.
- Lancer le logiciel « ispDesignExpert System » situé dans « Lattice Semiconductor ». Une fenêtre similaire à la figure 1 doit s'ouvrir.

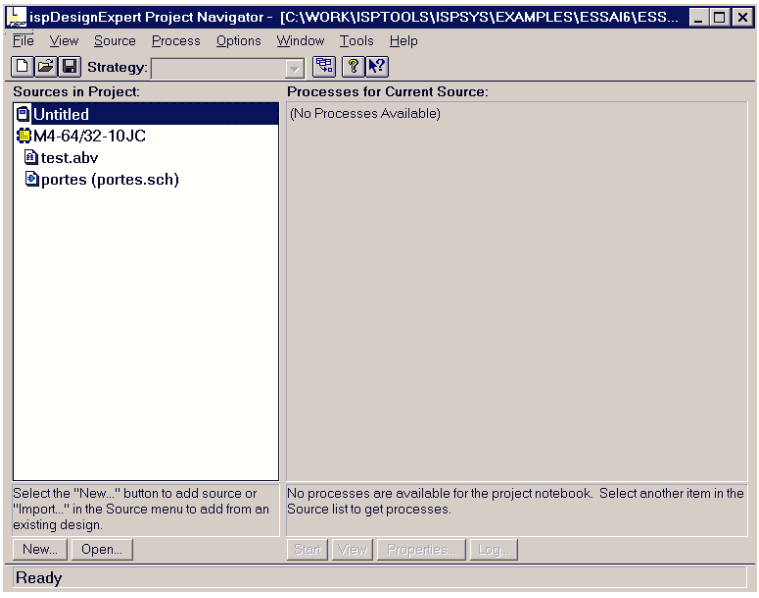



**Figure 1**

- Créer un nouveau projet appelé « Portes » en sélectionnant le menu « File » puis « New Project » du logiciel ispDesignExpert. Ce projet doit être créé dans le répertoire précédent. Laisser le type du projet sur « Schematic/ABEL ».

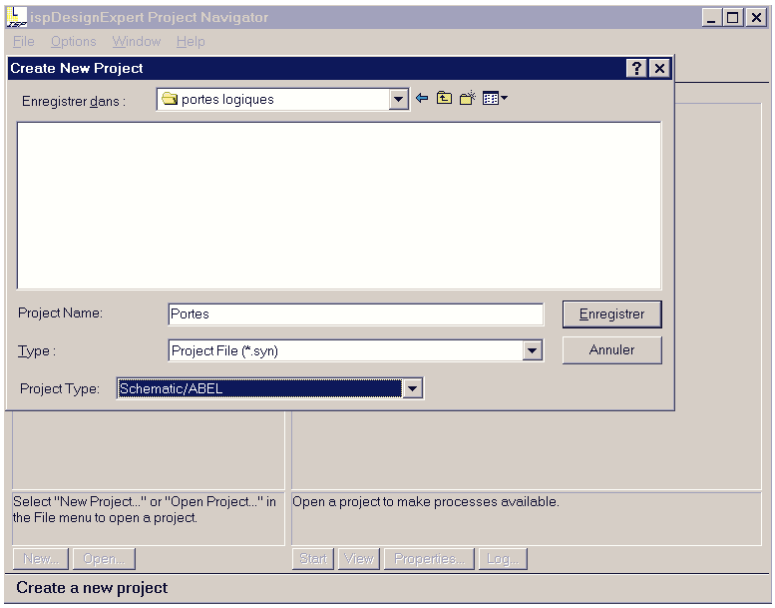



**Figure 2**

- La fenêtre suivante apparaît :

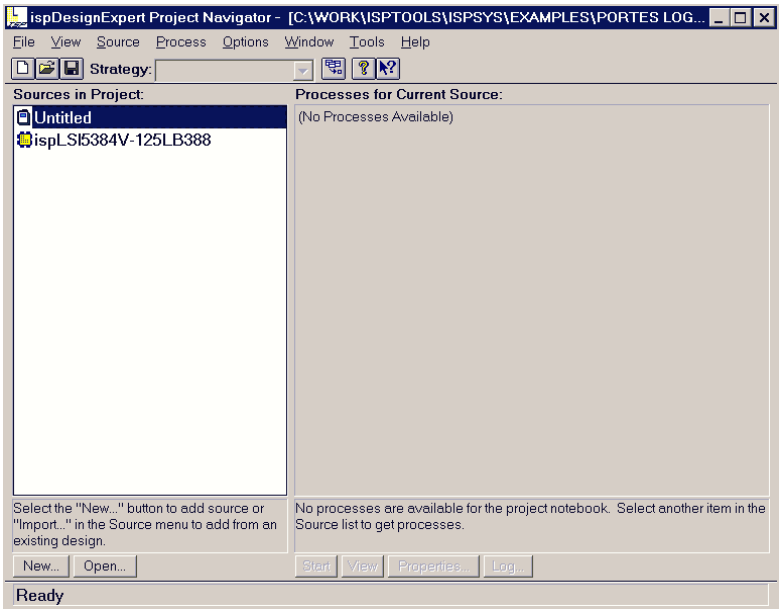



**Figure 3**

- En double cliquant sur « Untitled » dans la fenêtre hiérarchique, donner le titre « Test des portes logiques » à votre projet.

- En double cliquant sur ispLSI… dans la fenêtre hiérarchique, changer le PLD sélectionné. La fenêtre de sélection de la figure 4 apparaît. Choisir parmi les familles de PLD proposées le MACH 4, puis sélectionner le M4 64/32-15JC. Laisser les autres paramètres sur les valeurs par défaut (figure 4). Remarquer les informations indiquées par le logiciel dans le cadre « Device Information ».

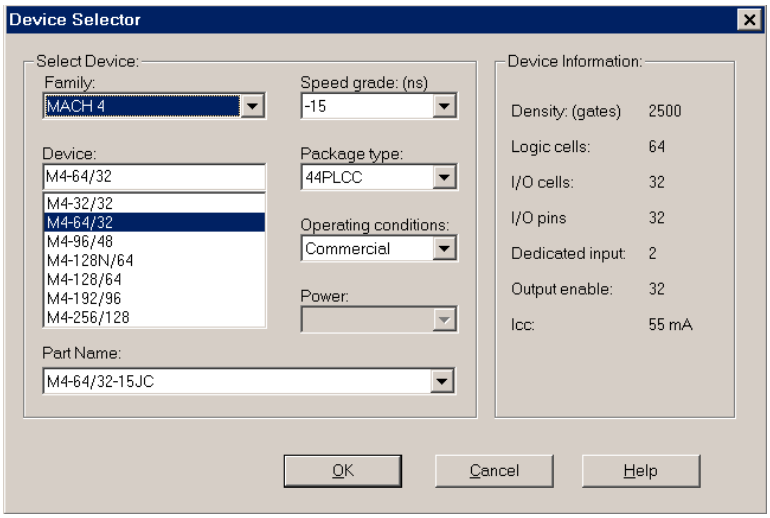



**Figure 4**

- Après sélection, la fenêtre de la figure 5 apparaît.

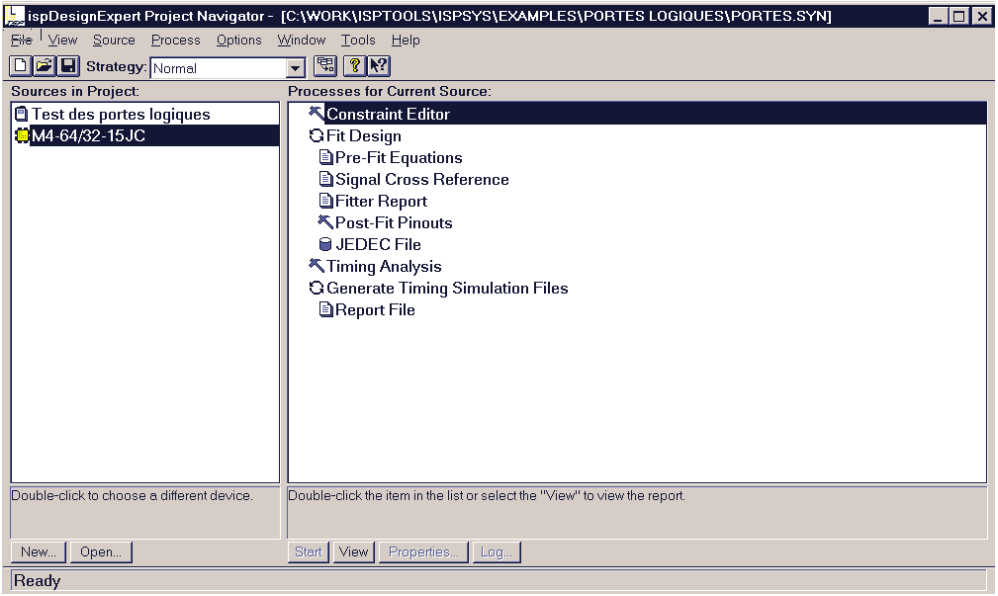



**Figure 5**

- La fenêtre hiérarchique contient actuellement le nom du projet et le CPLD sélectionné. Il s'agit maintenant de créer une « source ». La « source » est un fichier qui peut être de type schéma, ABEL, VHDL ou VERILOG. Ce fichier définit les fonctions logiques que l'ont veut programmer dans le CPLD. Aller dans le menu « Source » puis « New », la fenêtre suivante doit apparaître :

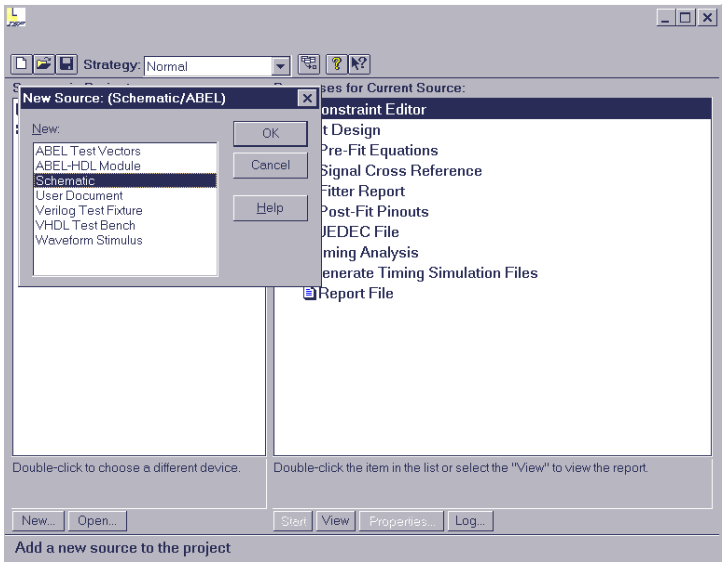



**Figure 6**

Remarquer que l'on peut également importer une source déjàexistante.

- On va créer nos portes logiques en utilisant un « Schematic ». La fenêtre de l'éditeur de schéma s'ouvre, puis entrer comme nom du schéma « Portes ». Important : le nom doit faire moins de 8 caractères et être en un seul mot.

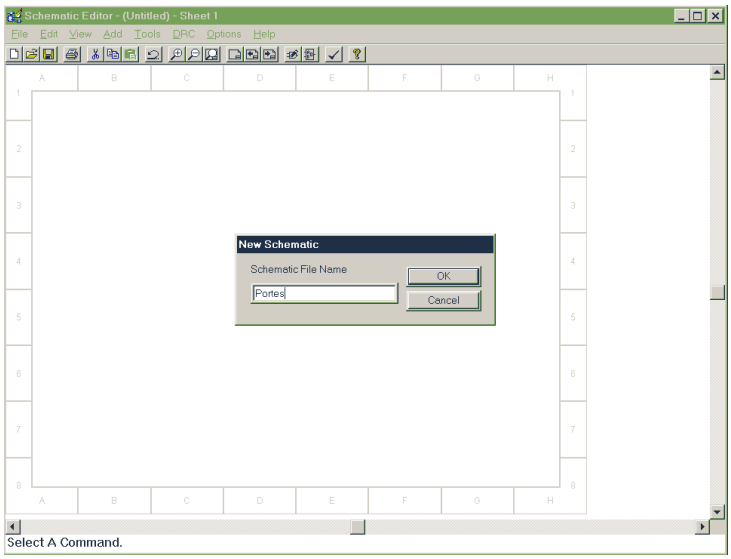



**Figure 7**

- Remarquer que le fichier « portes (portes.sch) » est apparu dans la fenêtre hiérarchique. Si vous fermez la fenêtre de l'éditeur de schéma, il est possible de la ré ouvrir en cliquant deux fois sur « portes (portes.sch) » dans la fenêtre hiérarchique.

- Dans la fenêtre de l'éditeur de schéma, il est possible d'agrandir le plan de travail. Aller dans « File », puis « Sheets », sélectionner « Resize » et choisir la dimension B.

- Nous allons placer quelques symboles sur le plan de travail. Cliquer sur l'icône « Add Symbol » de la barre flottante « Drawing ». La fenêtre de librairie de symboles s'ouvre et choisir dans la librairie générique « Gates.lib » une porte AND à deux entrées (G_2AND).


**Figure 8**

- Essayer de placer les portes suivantes sur le plan de travail :

**Figure 9**

Avec la barre flottante « Drawing », il est également possible de déplacer « Drag & Move » ou de supprimer « Delete » un symbole. Familiariser vous avec ces fonctions. Vous pouvez également déplacer ou supprimer un groupe de symboles.

- Il s'agit maintenant de câbler les portes logiques. En sélectionnant « Add Wire » dans la barre flottante « Drawing », tracer des fils comme représenté sur la figure 10.


**Figure 10**

Comme pour les symboles, il est possible de déplacer ou de supprimer des fils. Remarquer que les connexions entre plusieurs fils sont marquées par un carré bleu. Le croisement de deux fils sans carré bleu à l'intersection indique qu'ils ne sont pas interconnectés.

Avec la fonction « Undo » dans le menu « Edit », on peut retourner en arrière plusieurs fois dans le cas de mauvaises manipulations.

Avec les icônes « Loupe », vous pouvez zoomer ou dé zoomer sur le plan de travail.

Pensez àfaire des sauvegardes régulières avec l'icône « Save ».

- On va affecter des noms aux entrées et sorties du schéma.

**Figure 11**

Choisir « Add Net Name » dans le menu « Drawing ». Entrer le nom dans le champ en bas de la fenêtre. Vous pouvez placer le nom soit sur un fil, soit sur les extrémités d'un fil (représentées par des carrés rouges). Dans cet exemple, vous allez placer les noms uniquement sur les extrémités àla manière de la figure 12.

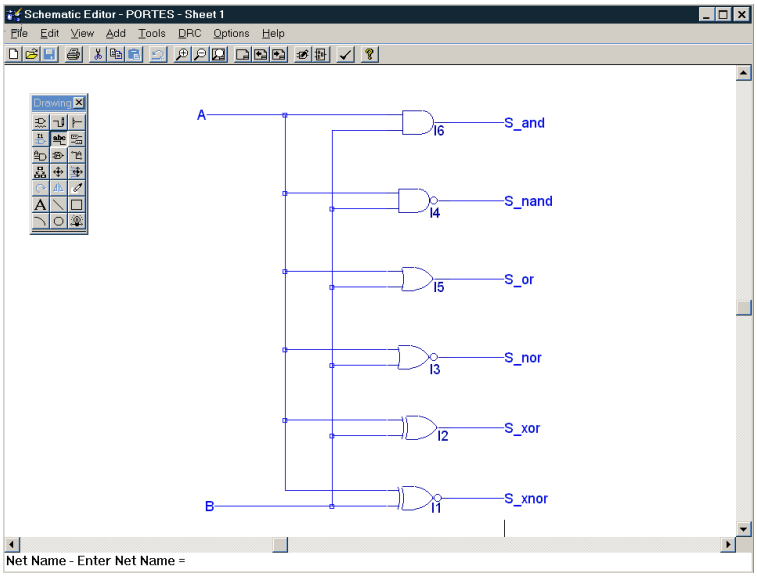



**Figure 12**

- Il faut indiquer au logiciel que A et B sont des entrées et que S_and, S_nand, S_or, S_nor, S_xor, S_xnor sont des sorties. Choisir « Add I/O Marker », cliquer sur « Input » puis sur le nom A. Faites en de même pour les autres noms. Le résultat final est indiqué sur la figure 13.

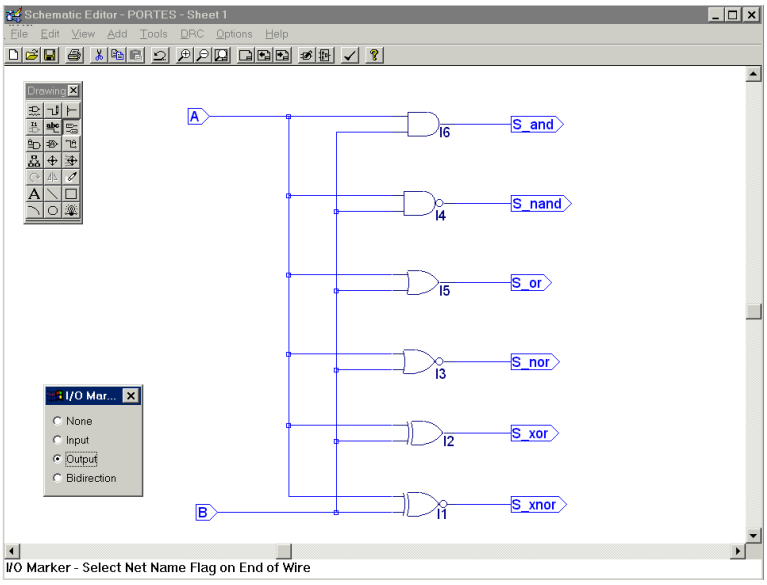



**Figure 13**

- L'éditeur de schéma permet de vérifier que le dessin ne contient pas d'erreur. Pour cela, aller dans « DRC » puis « Consistency Check ». La seule erreur rapportée devrait être « No Symbol for this Schematic ». En effet, il est possible (même si ce n'est pas indispensable) de générer un symbole avec comme entrées A et B et pour sorties les S_and, S_nand etc. Ce symbole pourrait être utilisé dans un autre schéma. Aller dans « File » puis « Matching Symbol » pour générer un symbole représentant le schéma. Si vous refaites un « DRC », il ne devrait plus y avoir d'erreur.
  On peut maintenant fermer l'éditeur de schéma.

- Dans la fenêtre hiérarchique, sélectionner « portes (portes.sch) » en cliquant une seule fois dessus. Dans la fenêtre de droite, vous voyez apparaître « Navigate Hierarchy », « Compile Schematic » et « Compiled Equations ». Faites une compilation de votre schéma en double cliquant sur « Compile Schematic ». Si une erreur apparaît, il faut modifier votre schéma. Faites un double clic sur « Compiled Equations » pour voir les équations générées par le compilateur (figure 14).

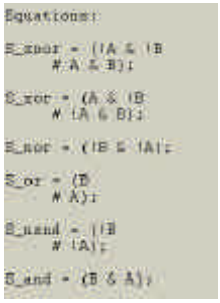



**Figure 14**

- Il faut vérifier que les sorties produisent bien les résultats attendus en fonction de l'état des entrées. Pour cela, il est nécessaire de créer un fichier de vecteurs de test.

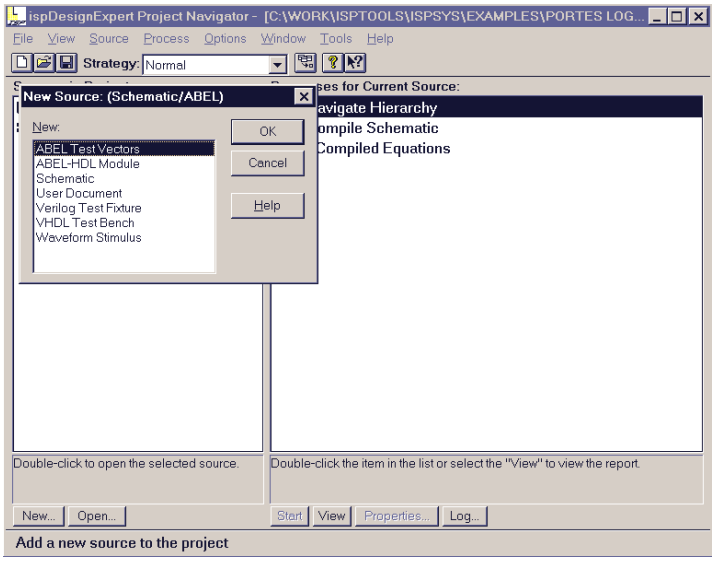



**Figure 15**

Aller dans « Source » puis « New » et choisir « ABEL test vectors ». Un éditeur de texte s'ouvre. Entrer comme nom « Stimuli ». Taper les lignes suivantes (langage ABEL-HDL) :

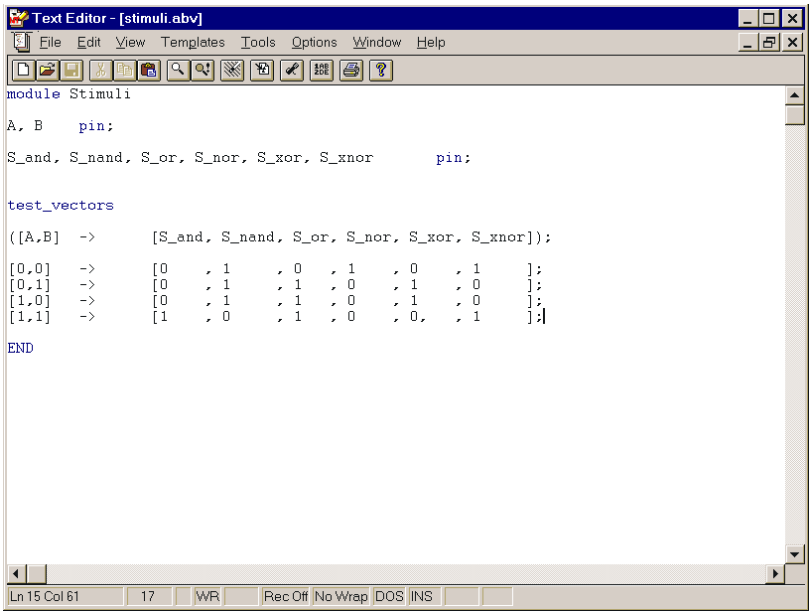



**Figure 16**

Sauvegarder puis fermer l'éditeur de texte.

- Dans la fenêtre hiérarchique, sélectionner « stimuli.abv » en cliquant une seule fois dessus. Dans la fenêtre de droite, compiler les vecteurs de test en double cliquant sur « Compile Test Vectors ». Si une erreur apparaît, il faut modifier le fichier de vecteurs de test. Faites une simulation des vecteurs de test en double cliquant sur « Simulate Compiled Equations ». En double cliquant sur « Equation Simulation Report », vous pouvez voir le résultat de la simulation (figure 17).

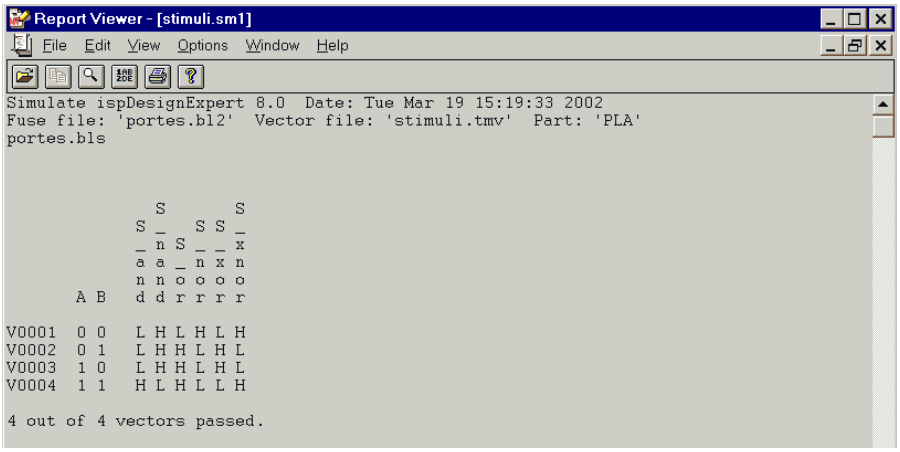



**Figure 17**

Si les résultats obtenus par simulation correspondent aux résultats attendus dans le fichier de vecteurs de test, le rapport indique « 4 out of  4 vectors passed ». Le circuit logique fonctionne donc correctement.

- Il est également possible de représenter les chronogrammes de fonctionnement du circuit logique. Faites un double clic sur « Equation Simulation Waveform », la fenêtre suivante s'ouvre :

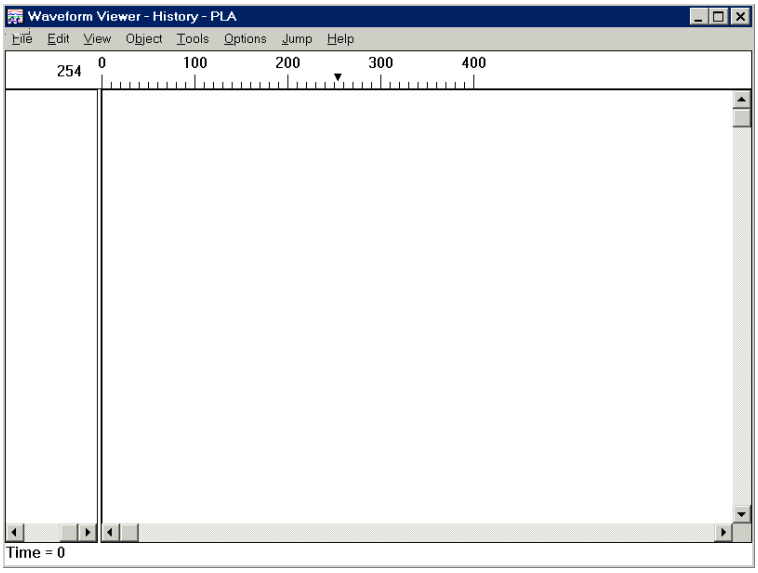



**Figure 18**

Aller dans « Edit » puis « Show ». Sélectionner les noms des nœuds que vous voulez faire apparaître sur le chronogramme.

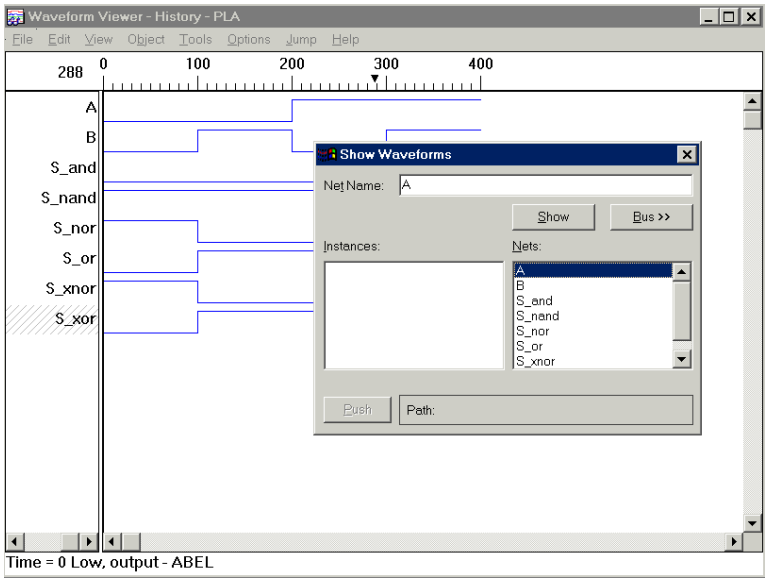



**Figure 19**

- « Simulate Pre-Fit Equations » permet de simuler les équations du circuit logique en tenant compte des caractéristiques du CPLD. Si par exemple le CPLD n'avait pas assez de portes logiques pour contenir le circuit logique, une erreur apparaîtrait à ce niveau.

- Sélectionner dans la fenêtre hiérarchique le CPLD « M4-64/32-15JC » en cliquant une seule fois dessus. Dans la fenêtre de droite, double cliquer sur le « Constraint Editor ». Cet éditeur permet d'associer aux noms A, B, S_and etc. un numéro de broche du boîtier du CPLD.

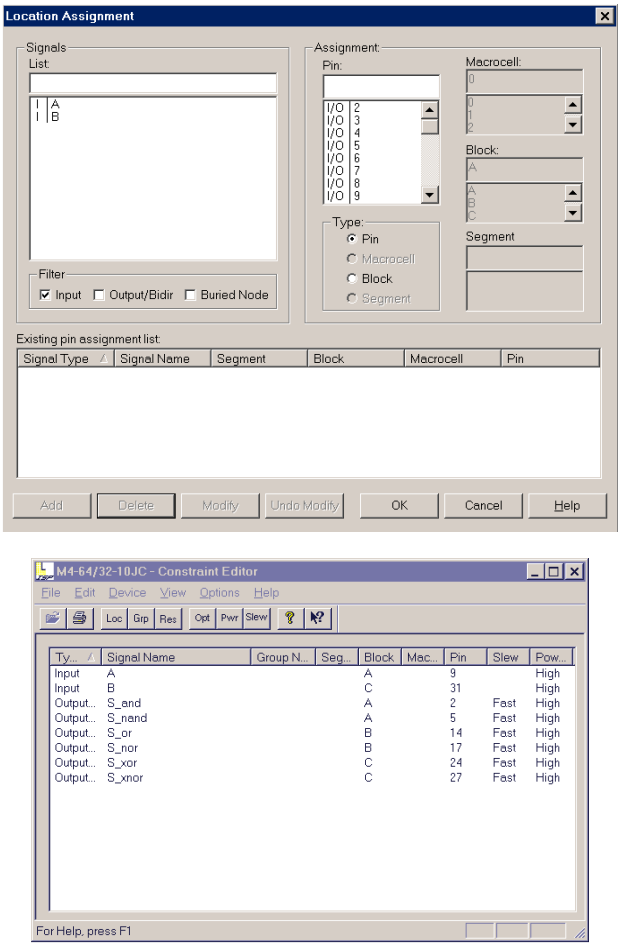



**Figure 20**

En cochant la case « Input » dans « Filter », il apparaît dans la liste toutes les entrées du circuit. On peut associer chaque entrée à une broche de la liste « Pin ». Chaque broche peut se comporter comme une entrée/sortie (I/O). Remarquer que les broches 11 et 33 sont réservées à l'horloge (C/l). En cochant la case « Output » dans « Filter », il apparaît dans la liste toutes les sorties du circuit. Il faut associer chaque sortie à une broche de la liste « Pin ».

Associer les broches comme il est indiqué sur la figure 20. Servez vous des indications qui sont données dans l'annexe pour savoir comment les broches du CPLD sont reliées à la platine. La platine est composée de 4 afficheurs à cristaux liquides et de 3 interrupteurs.

- Faites un double clic sur « Fit Design ». Le programmateur va vérifier que le circuit logique peut entrer dans le CPLD sélectionné. Le programmateur génère les équations du circuit logique « Pre-Fit Equations » ainsi que le fichier JEDEC. Le fichier JEDEC est un ensemble de données binaires indiquant les fusibles à griller dans le CPLD. En sélectionnant « JEDEC File » puis en cliquant sur « View » en bas de la fenêtre, il est possible de faire apparaître un rapport contenant le fichier JEDEC (figure 21).

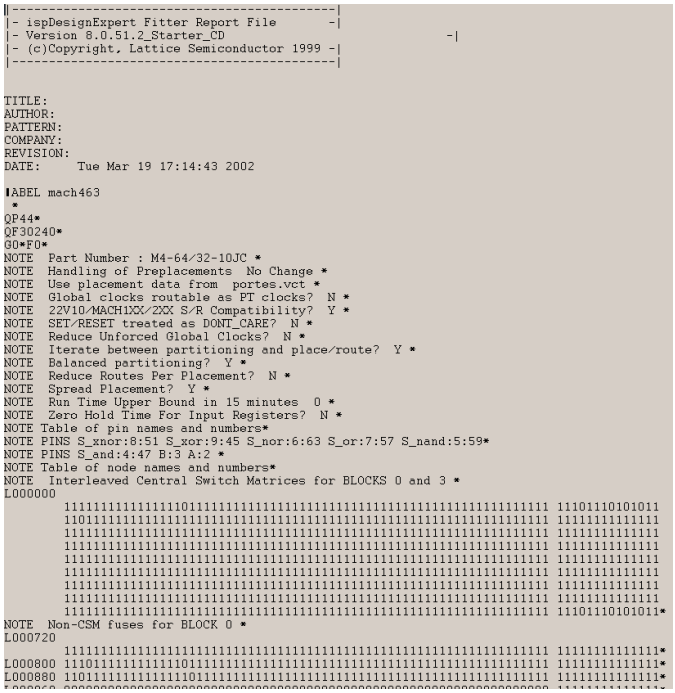



**Figure 21**

- Faites un double clic sur « Fitter Report » pour faire apparaître des informations sur l'utilisation du CPLD par le circuit logique.

- Faites un double clic sur « Timing Analysis » pour faire apparaître la figure 22.

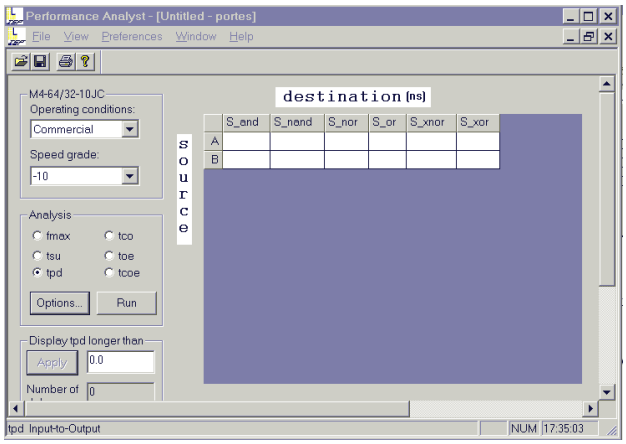



**Figure 22**

Il est uniquement possible de faire l'analyse du retard de propagation $t_{PD}$ (Propagation Delay) entre les sorties et les entrées. Cocher « $t_{PD}$ » puis faites « Run ». Le résultat est qu'il y a entre chaque entrée et sortie un retard de propagation de 15 ns. Remarquer que ce retard de propagation correspond au « Speed Grade » du CPLD sélectionné. Un M4-64/32-10JC a un retard de propagation de 10 ns tandis qu'un M4-64/32-12JC a un retard de propagation de 12 ns. Ce retard influe sur la fréquence maximale de fonctionnement du CPLD.

- Maintenant que le programme a été compilé, simulé et que le fichier JEDEC a été généré, il est temps de programmer le CPLD. Quitter le logiciel « ispDesignExpert » puis lancer le logiciel « ispVM System ». La fenêtre de la figure 23 apparaît.

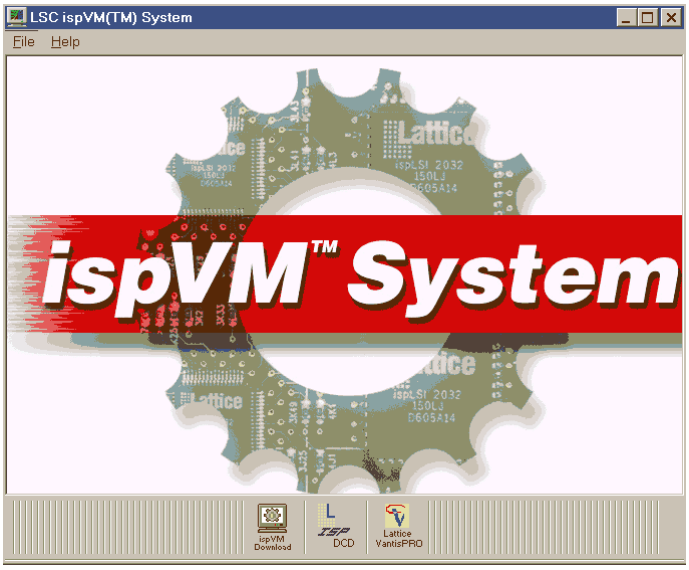



**Figure 23**

Le logiciel ispVM (in-system programmable Virtual Machine) est un programmateur de CPLD qui n'utilise pas les fichiers JEDEC, mais des fichiers au format VMF (Virtual Machine File). Les fichiers VMF, comme les fichiers JEDEC, contiennent la carte des fusibles à griller du CPLD. Mais les fichiers VMF contiennent en plus l'algorithme nécessaire pour programmer le CPLD. La programmation par fichiers VMF est plus rapide et plus universelle que le standard JEDEC.

Le programmateur ispVM convertit les fichiers JEDEC générés par ispDesignExpert en fichiers VMF.

- Cliquer sur l'icône « Lattice VantisPRO ». Une nouvelle fenêtre s'ouvre. Aller dans « File » puis « New ». Aller dans « Edit » puis « Add Device ». Cliquer sur « Select Part » et choisir le CPLD M4-64/32 indiqué sur la figure 24.

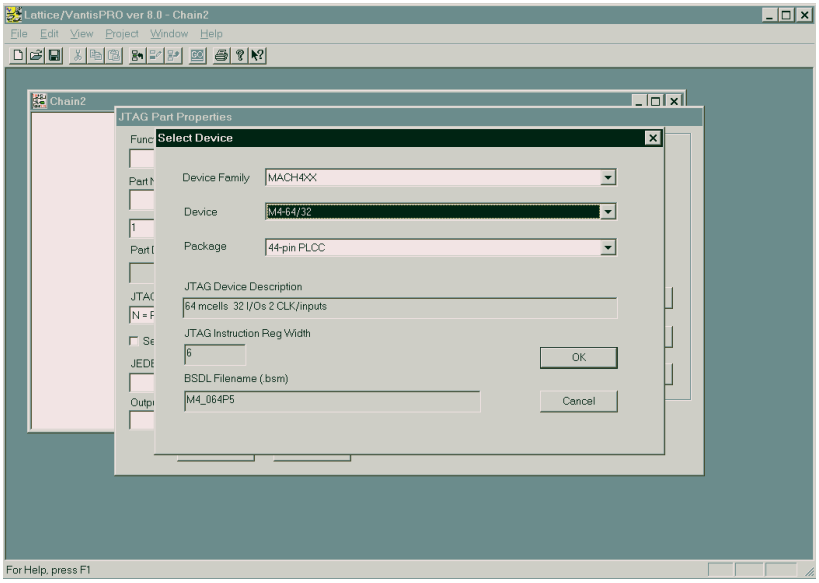



**Figure 24**

- Sélectionner l'opération « P = Erase, Program & Verify Device ». Laisser les autres champs comme sur la figure 25. Cliquer sur le bouton « Browse » pour spécifier le fichier JEDEC àutiliser « portes.jed ».

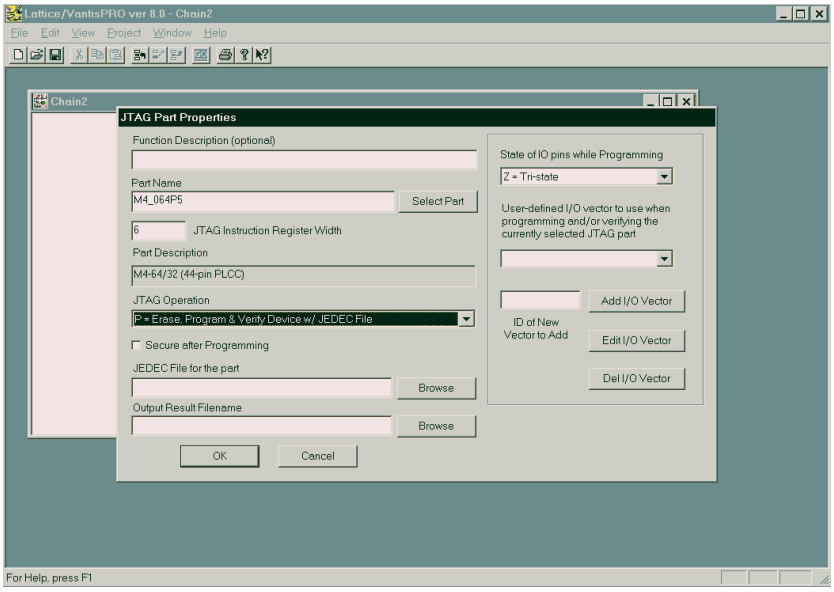



**Figure 25**

- Vous pouvez lancer la programmation du CPLD en cliquant sur l'icône « GO ». Après une vérification du bon fonctionnement du MACH avec les broches de test, la programmation est effectuée en quelques secondes.


**Figure 26**

- Une fois que le CPLD est programmé, vérifier le bon fonctionnement du circuit logique. Par exemple : la sortie de la porte AND a été fixée sur le segment (a) de l'afficheur U24 (voir annexe). Si on effectue aucune pression sur les deux interrupteurs (A = B = 1), la sortie de la porte AND est à l'état haut. Mais comme les LEDs de l'afficheur sont à anode commune, un état haut sur le segment (a) provoque son extinction. Si un des deux interrupteurs est pressé (A = 0 ou B = 0) ou si on presse simultanément les deux (A = B = 0), le segment (a) doit s'allumer. Le segment (d) a un comportement inverse du segment (a) étant donné qu'il est relié à la sortie de la porte NAND.

## *Questions*

Les réponses aux questions suivantes doivent être rédigées dans un rapport qui sera rendu à la fin du TP.

- Donner le pourcentage d'utilisation des broches d'entrées/sorties ainsi que le pourcentage d'utilisation en macrocellules.

- Les macrocellules ont elles un fonctionnement synchrone ou asynchrone ?

- Expliquer pourquoi le compilateur a écrit les équations des portes NAND et NOR sous la forme donnée en figure 14.

- Expliquer pourquoi toutes les portes ont le même retard de propagation.

- Modifier le circuit logique de telle manière qu'un interrupteur pressé ou qu'un segment allumé corresponde à un état logique haut. Programmer le CPLD.

## *Les portes logiques (description comportementale)*

Il s'agit de refaire le même exemple que précédemment mais en utilisant le langage de description comportementale ABEL.

- Créer à partir de l'explorateur de Windows un nouveau répertoire appelé « Portes Logiques 2 ».
- Créer un nouveau projet appelé « Portes » dans le répertoire précédent avec le logiciel ispDesignExpert.
- Donner le titre « Test des portes logiques » puis sélectionner le M4-64/32-15JC.
- Créer la source « ABEL-HDL Module » de la figure 27. Entrer pour les champs « Module Name » et « File Name » le nom « Portes ».

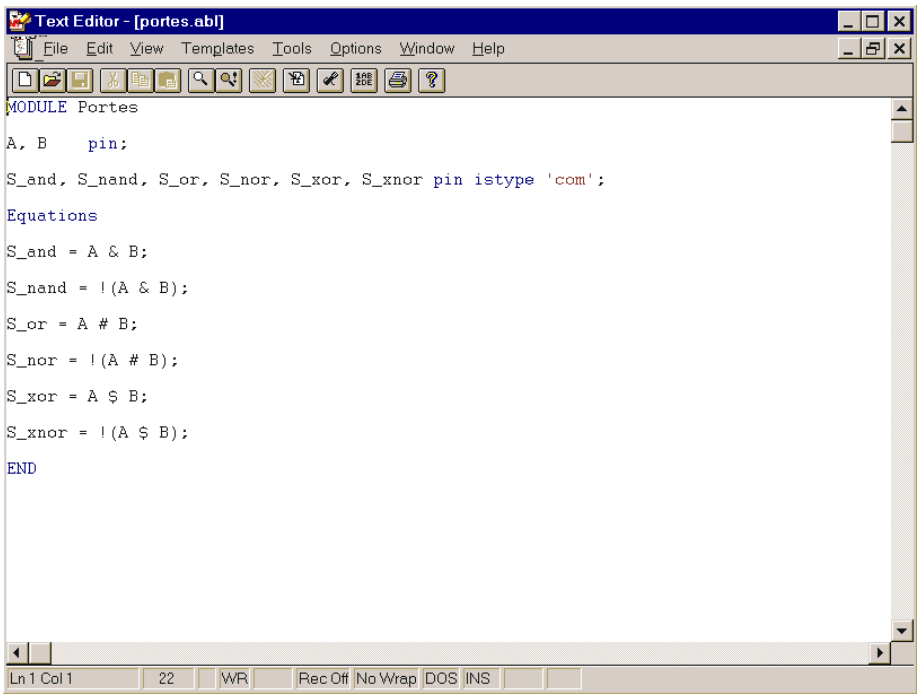



**Figure 27**

- Importer le fichier « Stimuli.abv » qui avait été créé dans l'exemple précédent.
- Vérifier que la simulation fonctionne correctement.
- Comparer les résultats obtenus de la description comportementale avec ceux de la description symbolique (équations, chronogrammes et retard de propagation).
- Associer les broches comme sur la figure 20, puis programmer le CPLD.

## *Compteur asynchrone 4 bits*

Dans cet exemple, il s'agit de programmer un compteur asynchrone modulo 10 en description structurelle.

- Créer un nouveau répertoire « Compteur Asynchrone ».
- Créer un nouveau projet « Compteur ».
- Donner comme titre « Compteur Asynchrone » et choisir le M4-64/32-15JC.
- Créer une nouvelle source « Schematic » de nom « Compteur ».
- Entrer le schéma suivant :

**Figure 28**

Les bascules sont dans la librairie « Regs.lib ». Le multiplexeur est dans la librairie « Muxes.lib ». Les VCC sont dans la librairie « Gates.lib ».

Ce compteur incrémente les sorties S sur le front descendant de l'horloge CLK. L'entrée RAZ permet la remise à zéro des sorties S de manière asynchrone. Ce compteur incrémente les sorties S entre 0 et 9. Lorsque l'état 9 est atteint, la sortie RCO est mise àl'état haut. Cette sortie peut servir d'horloge pour un deuxième compteur.

- Créer un fichier de vecteurs de test appelé « Stimuli » pour simuler le fonctionnement du compteur.

```
Module Stimuli

CLK, RAZ        pin;

S0, S1, S2, S3, RCO     pin;

S = [S0, S1, S2, S3];


Test_vectors

([CLK, RAZ]     ->      [ S , RCO])

 [.C.,  1 ]     ->      [.X., .X.];

@repeat 15  {[.C.,  0 ]   ->    [.X., .X.];}


END
```

**Figure 29**

Dans cet exemple, il n'est pas question de programmer le CPLD.

## *Questions*

Les réponses aux questions suivantes doivent être rédigées dans un rapport qui sera rendu àla fin du TP.

- Commenter les lignes du fichier de vecteurs de test.
- Donner les chronogrammes des entrées CLK et RAZ ainsi que des sorties S0, S1, S2, S3 et RCO.
- Donner les retards de propagation $t_{CO}$ (clock to output) entre l'horloge CLK et les sorties des bascules S0, S1, S2 et S3. Que constatez vous ?

## *Décodeur 7 segments*

Il s'agit de programmer un décodeur 7 segments en description comportementale.

- Créer un nouveau répertoire « Décodeur 7 segments ».
- Créer un nouveau projet « decodeur » (ne PAS mettre d'accent !).
- Donner comme titre « Décodeur 7 segments  » et choisir le M4-64/32-15JC.
- Créer une nouvelle source « ABEL-HDL Module ». Entrer pour les champs « Module Name » et « File Name » le nom « decodeur » (ne PAS mettre d'accent !).
- Entrer le programme suivant dans l'éditeur de texte :

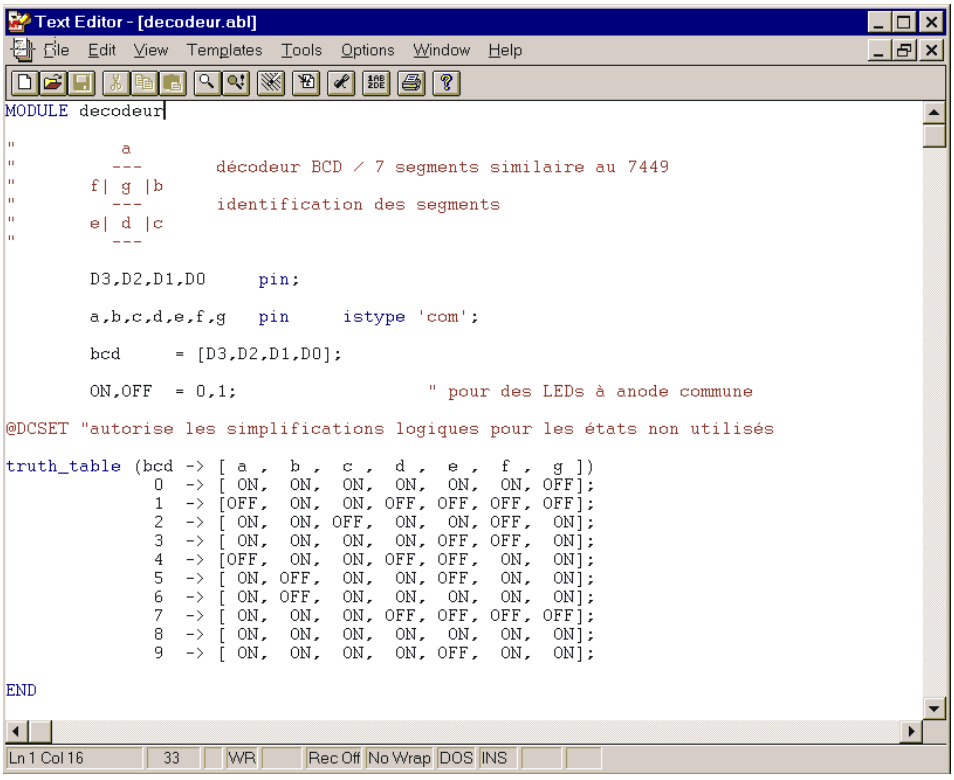



**Figure 30**

- Créer un fichier de vecteurs de test appelé « Stimuli » pour simuler le fonctionnement du décodeur.

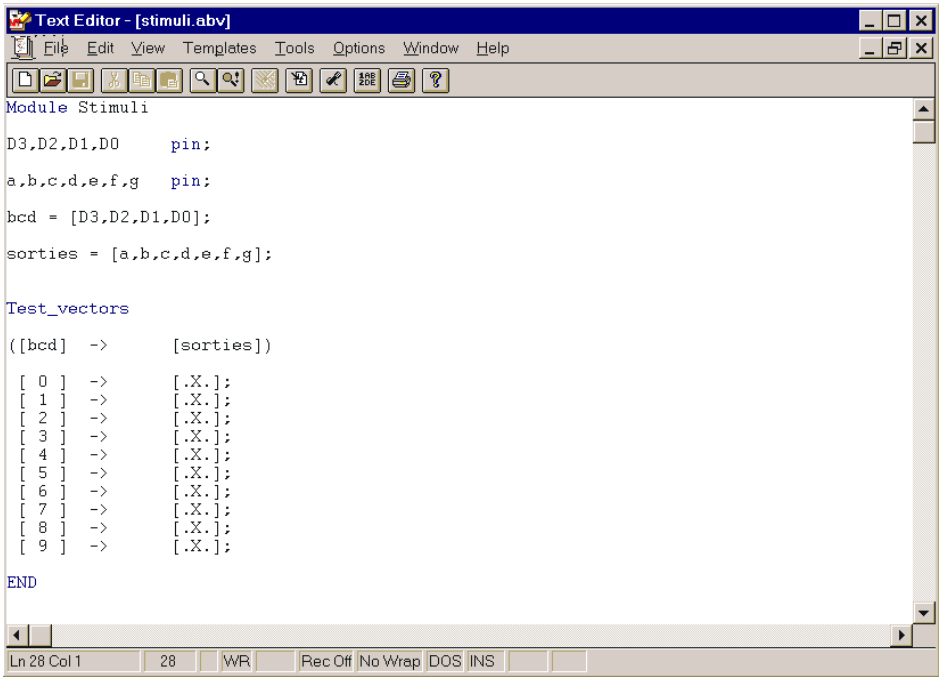



**Figure 31**

Dans cet exemple, il n'est pas question de programmer le CPLD.

## *Questions*

Les réponses aux questions suivantes doivent être rédigées dans un rapport qui sera rendu àla fin du TP.

- Commenter les lignes du fichier du décodeur.
- Commenter les lignes du fichier des vecteurs de test.
- Représenter les chronogrammes D0, D1, D2, D3 et a, b, c, d, e, f, g. Vérifier que l'état des segments (on/off) pour chaque valeur entre 0 et 9 est correct.
- Dans quel état (on/off) sont les segments pour les valeurs interdites 10 à15 ?

## *Affichage de la valeur d'un compteur par décodage*

- Créer un nouveau répertoire « Compteur décodeur ».
- Créer un nouveau projet « Comdec ».
- Donner comme titre « Compteur décodeur » et choisir le M4-64/32-15JC.
- Importer le fichier du compteur « compteur.sch » ainsi que le fichier du décodeur 7 segments « decodeur.abl ».
- Créer une nouvelle source « Schematic » de nom « comdec ».
- Si ce n'est pas déjàfait, générer un symbole pour le compteur.
- Pour utiliser dans le schéma « comdec » le décodeur, il est nécessaire de lui générer un symbole. Dans l'éditeur de schéma, aller sur « File » puis « Generate Symbol ». Aller dans le répertoire « Compteur décodeur » et choisir le fichier « decodeur.naf ». Un message apparaît indiquant que le symbole vient d'être généré.
- Entrer le schéma suivant dans « comdec » :

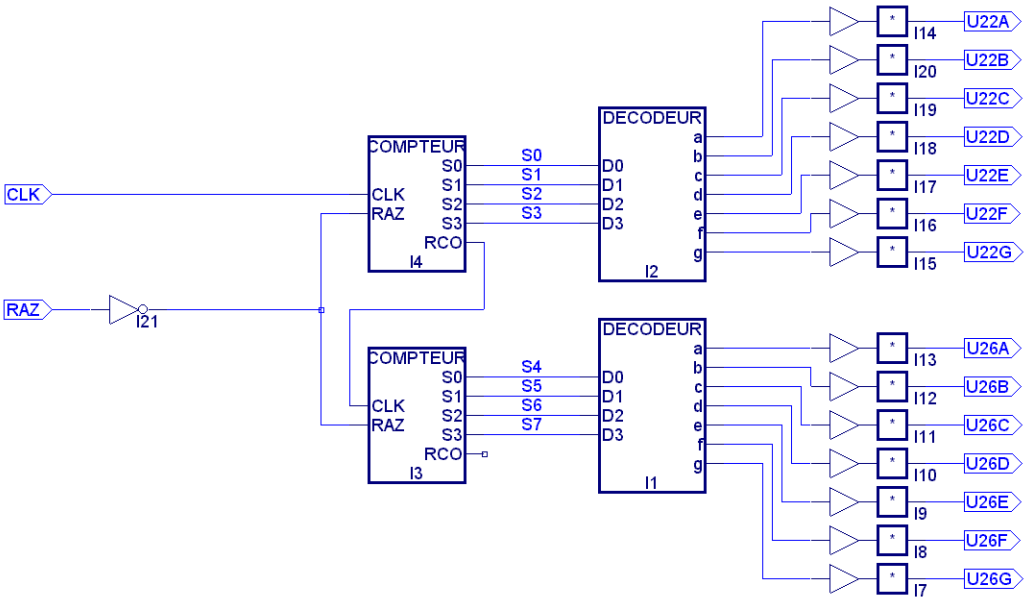


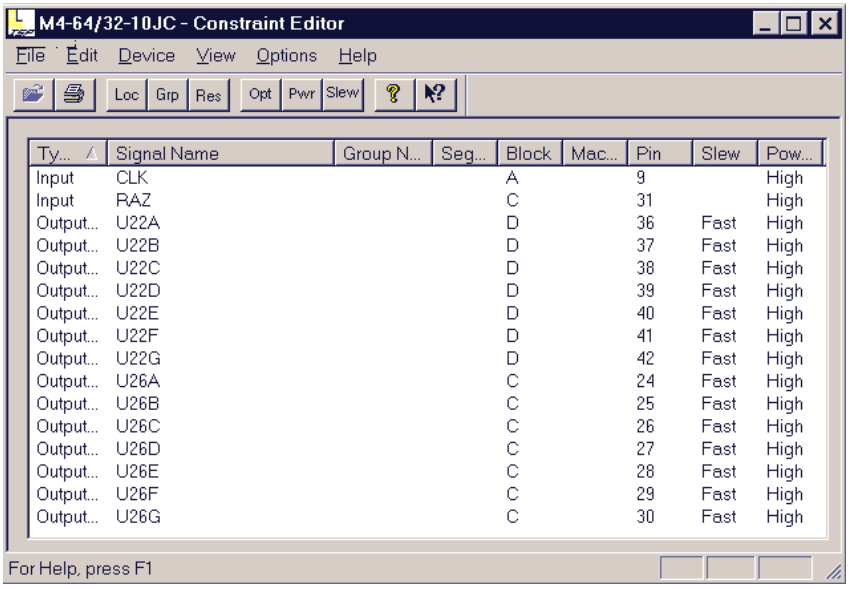

**Figure 32**

- Créer un fichier de vecteurs de test appelé « Stimuli » pour simuler le fonctionnement du compteur décodeur.

```
Module Stimuli

CLK, RAZ        pin;

U22A,U22B,U22C,U22D,U22E,U22F,U22G      pin;

U26A,U26B,U26C,U26D,U26E,U26F,U26G      pin;

S22 = [U22A,U22B,U22C,U22D,U22E,U22F,U22G];

S26 = [U26A,U26B,U26C,U26D,U26E,U26F,U26G];

Test_vectors

([CLK, RAZ]     ->      [S22 , S26])

 [.C.,  0 ]     ->      [.X., .X.];

@repeat 30  {[.C.,  1 ]   ->    [.X., .X.];}

END
```

**Figure 33**

- Associer les noms des entrées/sorties avec les numéros de broches :




| Ty... | Signal Name | Group N... | Seg... | Block | Mac... | Pin | Slew | Pow... |
|---|---|---|---|---|---|---|---|---|
| Input | CLK | | | A | | 9 | | High |
| Input | RAZ | | | C | | 31 | | High |
| Output... | U22A | | | D | | 36 | Fast | High |
| Output... | U22B | | | D | | 37 | Fast | High |
| Output... | U22C | | | D | | 38 | Fast | High |
| Output... | U22D | | | D | | 39 | Fast | High |
| Output... | U22E | | | D | | 40 | Fast | High |
| Output... | U22F | | | D | | 41 | Fast | High |
| Output... | U22G | | | D | | 42 | Fast | High |
| Output... | U26A | | | C | | 24 | Fast | High |
| Output... | U26B | | | C | | 25 | Fast | High |
| Output... | U26C | | | C | | 26 | Fast | High |
| Output... | U26D | | | C | | 27 | Fast | High |
| Output... | U26E | | | C | | 28 | Fast | High |
| Output... | U26F | | | C | | 29 | Fast | High |
| Output... | U26G | | | C | | 30 | Fast | High |

**Figure 34**

Le bouton SW1 va servir d'horloge pour incrémenter les compteurs. Le bouton SW3 permet de remettre les compteurs àzéro.

- Compiler les équations, générer le fichier JEDEC et programmer le CPLD.

## *Questions*

Les réponses aux questions suivantes doivent être rédigées dans un rapport qui sera rendu àla fin du TP.

- Représentez les chronogrammes de simulation CLK, RAZ, U22A…U22G, U26A…U26G, et S0…S7.
- Que constatez vous de particulier sur les chronogrammes de simulation U22A…U22G ?
- Comment pouvez vous expliquer ce phénomène ?
- Proposer une solution pour régler ce problème (donner uniquement la solution, il ne s'agit pas de simuler ou de câbler cette solution).
- En pratique, est-ce que ce problème est visible sur les afficheurs ? Expliquer pourquoi.

Lors du fonctionnement, on se rend compte d'un autre problème que la simulation ne montre pas, mais qui est nettement plus gênant. L'interrupteur utilisé pour incrémenter le compteur a tendance à rebondir. Le compteur est parfois incrémenté plusieurs fois alors qu'une seule pression semble avoir été effectuée sur l'interrupteur. Il faut rajouter le circuit anti-rebond de la figure 35 entre l'interrupteur et l'entrée d'horloge du compteur.

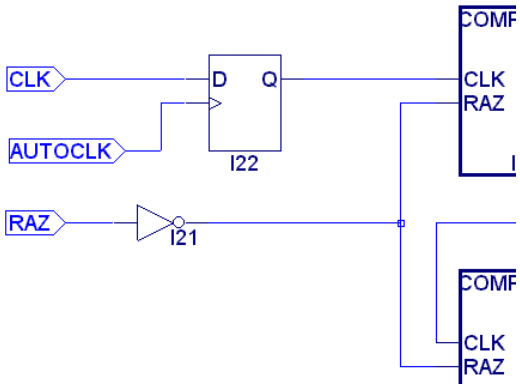



**Figure 35**

L'entrée AUTOCLK doit être associé àla broche 11 (horloge de fréquence ajustable). Régler la fréquence de cette horloge sur 64 Hz. Compiler et programmer le CPLD. Vérifier que les rebonds de l'interrupteur ont effectivement été supprimés.

## *Défilement d'un texte sur les afficheurs 7 segments*

Il s'agit de faire déplacer le mot « bonjour » de la droite vers la gauche sur les 4 afficheurs 7 segments. Pour cela, on a besoin d'une mémoire qui contient le mot « bonjour », de registres àdécalage pour déplacer le mot, et de décodeurs 7 segments pour l'affichage.

- Créer un nouveau répertoire « Défilement ».
- Créer un nouveau projet « Scrolling ».
- Donner comme titre « Défilement d'un texte » et choisir le M4-64/32-15JC.

## *Décodeur 7 segments*

- Créer une source ABEL de nom « decodeur » (ne PAS mettre d'accent !).
- Entrer le programme suivant :

```
MODULE decodeur

INTERFACE (D2, D1, D0 -> a, b, c, d, e, f, g);

TITLE 'decodeur 7 segments'

"          a
"         ---
"       f| g |b
"         ---
"       e| d |c
"         ---

        D2,D1,D0        pin;
        a,b,c,d,e,f,g   pin     istype 'com';
        data    = [D2,D1,D0];
        led     = [a,b,c,d,e,f,g];
        ON,OFF  = 0,1;                  " pour des LEDs à anode commune


truth_table (data -> [ a ,  b ,  c ,  d ,  e ,  f ,  g ])
              0   -> [OFF, OFF, OFF, OFF, OFF, OFF, OFF];
              1   -> [OFF, OFF,  ON,  ON,  ON,  ON,  ON];       " b
              2   -> [OFF, OFF,  ON,  ON,  ON, OFF,  ON];       " o
              3   -> [OFF, OFF,  ON, OFF,  ON, OFF,  ON];       " n
              4   -> [OFF,  ON,  ON,  ON,  ON, OFF, OFF];       " j
              5   -> [OFF, OFF,  ON,  ON,  ON, OFF, OFF];       " u
              6   -> [OFF, OFF, OFF, OFF,  ON, OFF,  ON];       " r

END
```

**Figure 36**

Le décodeur a 3 entrées binaires et 7 sorties. Le « b » est codé avec la valeur 1, le « o » est codé avec la valeur 2, etc. Les lettres du mot « bonjour » sont affichées en minuscules sur les afficheurs.

## *Registre à décalage*

- Créer une source « Schematic » de nom « registre ».

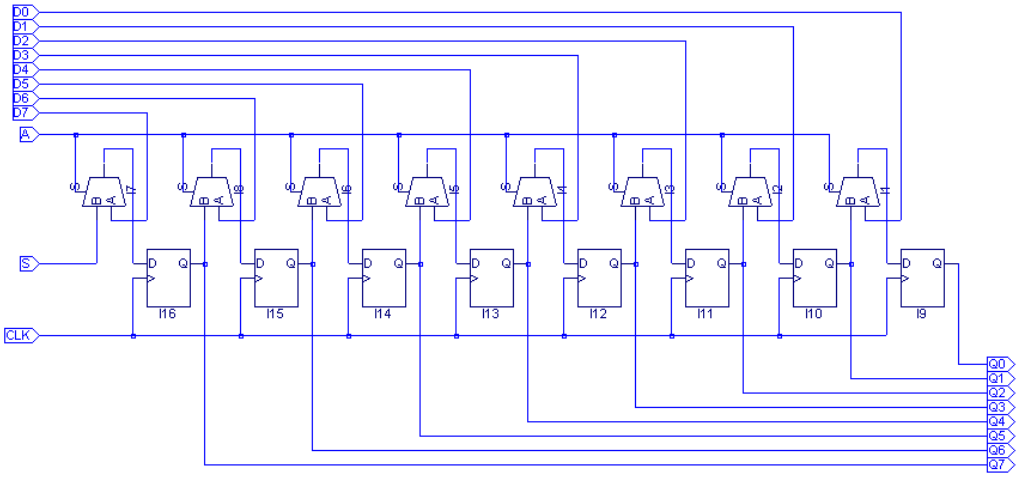



**Figure 37**

C'est un registre à décalage 8 bits. Les données sur les sorties sont décalées de Q7 vers Q0 à chaque front montant de l'horloge CLK. Les données peuvent être chargées en série avec l'entrée S ou en parallèle avec les entrées D7 à D0. L'entrée A permet de sélectionner le mode de fonctionnement du registre, il y a chargement parallèle si A = 0 ou décalage avec chargement série si A = 1.

## *Mémoire*

- Créer une source ABEL de nom « memoire » (ne PAS mettre d'accent !).

```
MODULE memoire

cs      pin;
d00..d07        pin istype 'com';
d10..d17        pin istype 'com';
d20..d27        pin istype 'com';
d2 = [d20..d27];
d1 = [d10..d17];
d0 = [d00..d07];

equations

when (cs == 1) then
" valeur dec = 1 2 3 4 2 5 6 0
"              b o n j o u r
{[d00..d07] = [1,0,1,0,0,1,0,0];
 [d10..d17] = [0,1,1,0,1,0,1,0];
 [d20..d27] = [0,0,0,1,0,1,1,0];}

else when (cs == 0) then
{d2 = 0;
 d1 = 0;
 d0 = 0;}

END
```

**Figure 38**

C’est une mémoire de 8 mots de 3 bits avec une entrée de sélection CS et 8 * 3 sorties. Le mot de 3 bits numéro 0 se trouve sur les sorties d20 (poids fort) à d00 (poids faible). Le mot de 3 bits numéro 1 est sur les sorties d21 à d01 etc.

Le mot numéro 0 est chargé avec la valeur décimale 1 qui correspond après décodage à la lettre « b ». Le mot numéro 1 est chargé avec la valeur 2 qui correspond à la lettre « o » etc. Les lettres « bonjour » sont ainsi inscrites dans la mémoire.

## *Schéma complet pour le défilement d’un texte*

- Créer une source « Schematic » de nom « scrolling ».
- Penser à générer les symboles pour le décodeur et la mémoire.

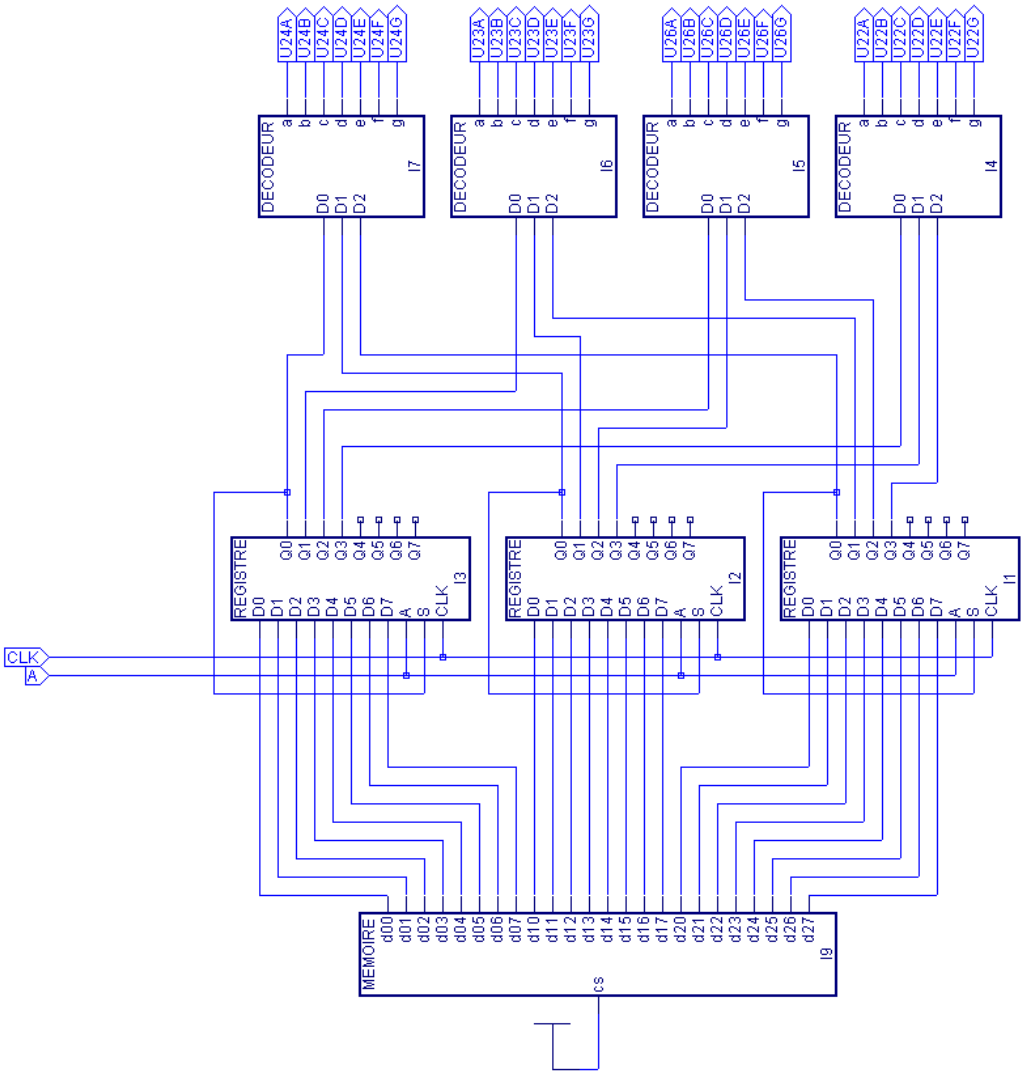



**Figure 39**

## *Questions*

- Si A = 0 et qu'un front montant est appliqué sur CLK, qu'apparaît-il sur les 4 afficheurs ?

- On considère que le circuit est dans l'état précédent. On met A = 1 et on applique un nouveau front montant sur CLK, qu'apparaît-il sur les 4 afficheurs ?

- Pourquoi la sortie Q0 de chaque registre est reliée àl'entrée S ?

## *Programmation*

- Associer les noms des entrées/sorties avec les numéros de broches :

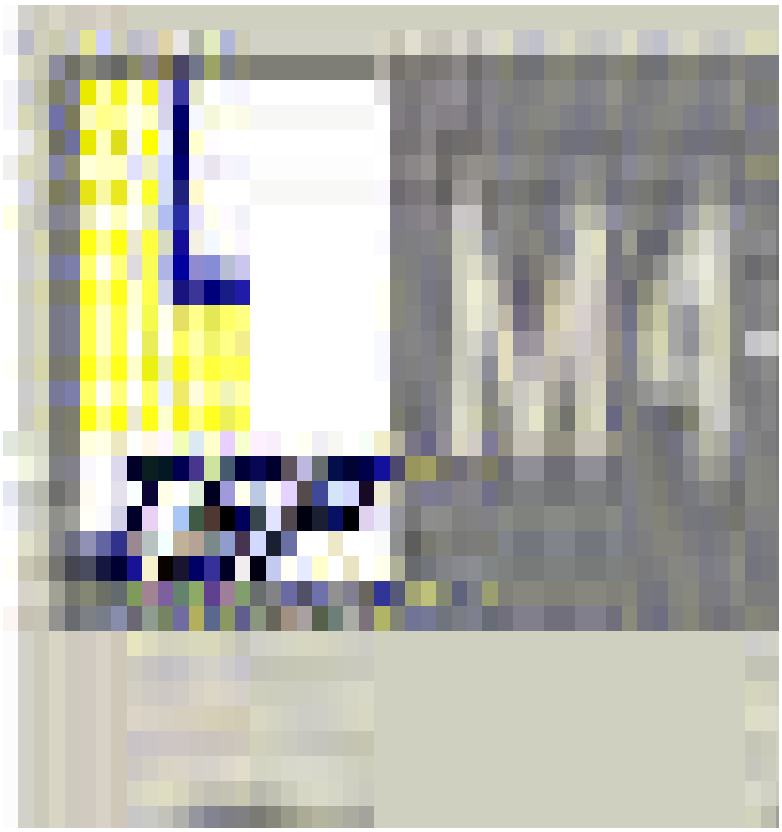


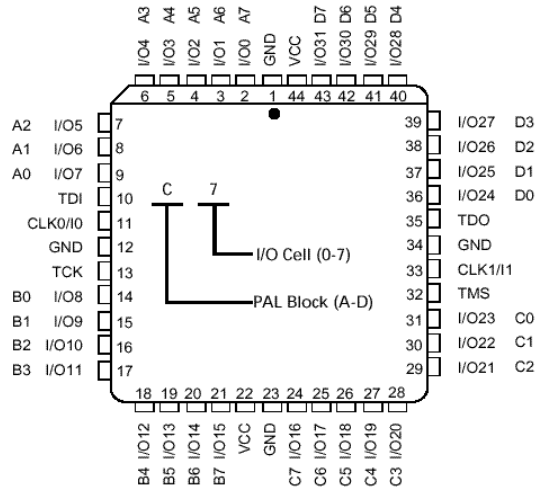
**Figure 40**

L'interrupteur SW1 permet d'initialiser les registres àdécalage. Régler la fréquence de l'horloge sur 2 Hz.

- Générer le fichier JEDEC puis programmer le CPLD.

## *ANNEXES*

- CPLD MACH4-64/32 en boîtier PLCC (Plastic Leaded Chip Carrier) :




CLK/I = Clock or Input
GND = Ground
I/O = Input/Output
$V_{CC}$ = Supply Voltage
TDI = Test Data In
TCK = Test Clock
TMS = Test Mode Select
TDO = Test Data Out

- Architecture du CPLD :

- Liste des broches de la platine :

**Pin List for the ISP Board**

| Device Pin | Pin Definition | MACH Inputs | LED | Comment |
|---|---|---|---|---|
| 1 | GND | | | |
| 2 | I/O 0 | | U24-A | |
| 3 | I/O 1 | | U24-B | |
| 4 | I/O 2 | | U24-C | |
| 5 | I/O 3 | | U24-D | |
| 6 | I/O 4 | | U24-E | |
| 7 | I/O 5 | | U24-F | |
| 8 | I/O 6 | | U24-G | |
| 9 | I/O 7 | SW1 | U24-H (DP) | **Use as Input only** |
| 10 | TDI | | | |
| 11 | CLK 0 / I 0 | CK0 Clock | | Select with jumper |
| 12 | GND | | | |
| 13 | TCK | | | |
| 14 | I/O 8 | | U23-A | |
| 15 | I/O 9 | | U23-B | |
| 16 | I/O 10 | | U23-C | |
| 17 | I/O 11 | | U23-D | |
| 18 | I/O 12 | | U23-E | |
| 19 | I/O 13 | | U23-F | |
| 20 | I/O 14 | | U23-G | |
| 21 | I/O 15 | SW2 | U23-H (DP) | **Use as Input only** |
| 22 | VCC | | | |
| 23 | GND | | | |
| 24 | I/O 16 | | U26-A | |
| 25 | I/O 17 | | U26-B | |
| 26 | I/O 18 | | U26-C | |
| 27 | I/O 19 | | U26-D | |
| 28 | I/O 20 | | U26-E | |
| 29 | I/O 21 | | U26-F | |
| 30 | I/O 22 | | U26-G | |
| 31 | I/O 23 | SW3 | U26-H (DP) | **Use as Input only** |
| 32 | TMS | | | |
| 33 | CLK 1 / I 1 | CK1 Clock | | 4 Hz Clock signal |
| 34 | GND | | | |
| 35 | TDO | | | |
| 36 | I/O 24 | | U22-A | |
| 37 | I/O 25 | | U22-B | |
| 38 | I/O 26 | | U22-C | |
| 39 | I/O 27 | | U22-D | |
| 40 | I/O 28 | | U22-E | |
| 41 | I/O 29 | | U22-F | |
| 42 | I/O 30 | | U22-G | |
| 43 | I/O 31 | | U22-H (DP) | |
| 44 | VCC | | | |

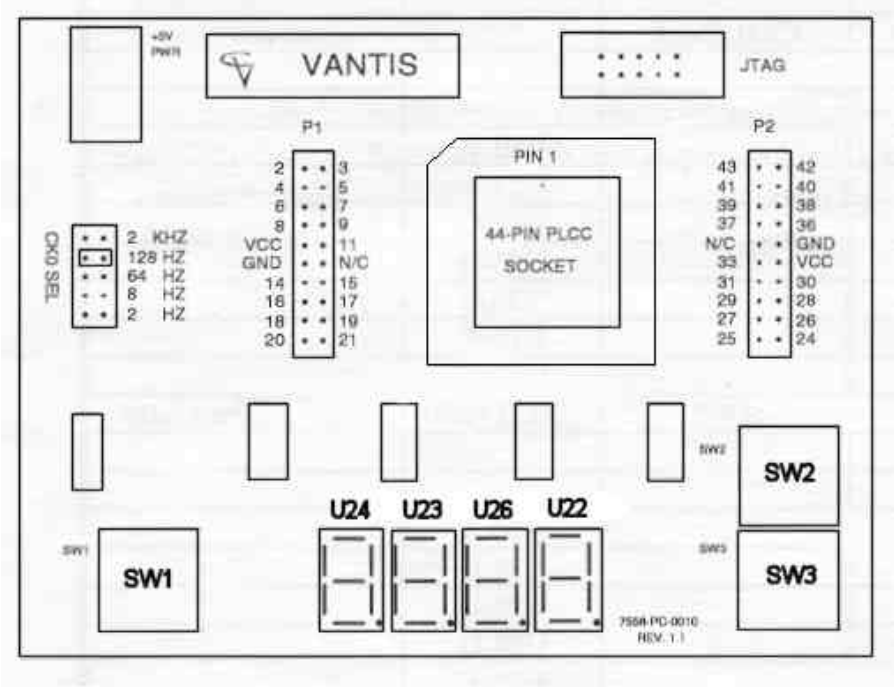
SW1, SW2 et SW3 sont les 3 interrupteurs de la platine. Lorsqu'on effectue une pression sur un interrupteur, l'état logique de la broche associée àl'interrupteur passe àl'état BAS.

La platine dispose de 4 afficheurs àcristaux liquides notés U22, U23, U24 et U25.

Les 7 segments de chaque afficheur sont notés de A à G. Un segment est allumé quand le niveau logique de la broche correspondante est àl'état BAS.

Le CPLD a deux horloges (broches 11 et 33). Seule la fréquence de l'horloge de la broche 11 peut être modifiée via une sélection par jumper.

- Platine :




- Afficheur 7 segments :