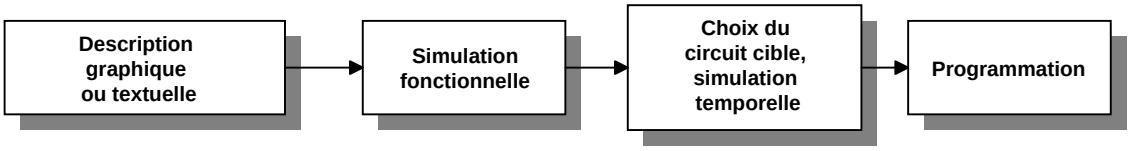
Travaux pratiques : programmation de CPLD et FPGA, première approche du logiciel Max+plus II

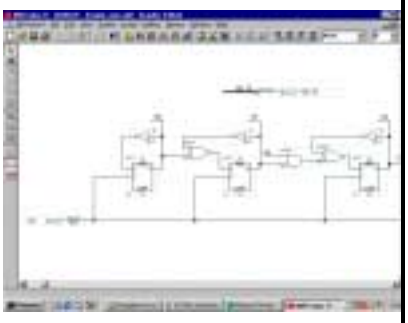
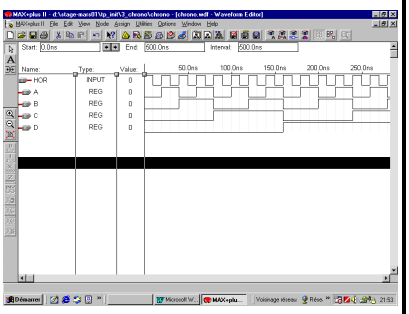
L'objectif de cette séance est de se familiariser avec les outils de programmation des composants logiques programmables. Nous utiliserons pour cela celui de la firme Altera : le logiciel Max+plus II (version 9.3), associé à une carte de développement.
Nous synthétiserons sur un circuit cible, une fonction permettant à partir du signal de l'oscillateur à quartz de la carte, de visualiser l'écoulement des secondes sur un des afficheurs de la carte. Nous aurons besoins pour cela d'implanter un diviseur de fréquence, un compteur et un décodeur BCD/7 segments.

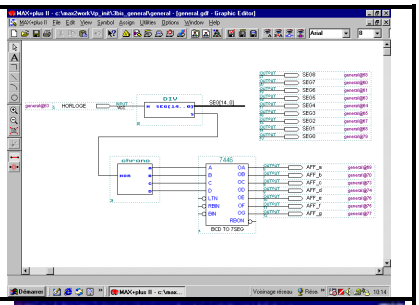
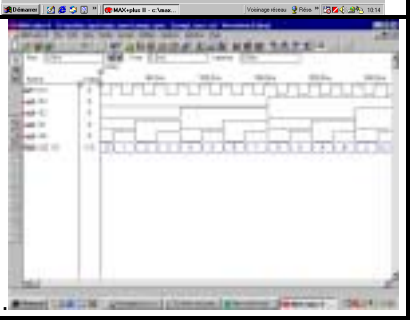
1. Présentation du logiciel

Il permet entre autres, la description d'un projet, sa compilation, sa simulation logique et temporelle, son analyse temporelle et la programmation d'un circuit CPLD ou FPGA, dit circuit cible.



La description du projet peut être faite à l'aide d'une des entrées suivantes :

		
- éditeur de texte pour l'utilisation du langage AHDL (langage propriétaire Altera), VHDL ou VERILOG.	- éditeur graphique qui permet d'utiliser les composants prédéfinis des bibliothèques fournies par le logiciel ou créées par l'utilisateur.	- éditeur de chronogrammes avec lequel on représentera l'évolution temporelle des entrées ainsi que celle attendue des sorties.

A chaque description est associé un symbole graphique du composant ainsi réalisé. L'éditeur graphique permettra alors de relier éventuellement ces composants les uns aux autres. On décrira ainsi un projet global comprenant des sous-ensembles édités indifféremment par une des trois entrées.	
Chaque sous-ensemble puis le projet global est ensuite compilé, puis simulé par le simulateur logique, puis analysé en fonction de critères temporels et envoyé vers le circuit cible par le programmateur	

Le circuit cible peut être choisi par l'utilisateur ou laissé au libre choix du logiciel en fonction de critères prédéfinis (optimisation de la vitesse ou du taux d'occupation du circuit).

Le déroulement de toutes ces opérations nécessite la génération par le logiciel d'un nombre important de fichier. Tous les fichiers relatifs à une description portent le même nom, mais une extension différente (voir en annexe les principales extensions).
Pour éviter les problèmes, il faudra faire preuve d'une certaine rigueur **et ne pas donner le même nom à deux descriptions différentes**.
Le logiciel ne traitant qu'un projet à la fois, il **faudra vérifier que le projet en cours de traitement est bien celui désiré, l'affichage se faisant en haut de l'écran. Il est conseillé de systématiquement déclarer la fenêtre active comme contenant le projet en cours (par un « Ctrl Shift J » -voir plus loin-) avant toute sauvegarde et compilation.**

Une aide pour chaque point particulier du logiciel est disponible en appuyant sur la touche **F1**.ou en cliquant sur l'icône **?** puis sur « le point mystérieux ».

Les menus disponibles dans le logiciel dépendent de la fenêtre dans laquelle vous vous trouvez : les options des éditeurs ne sont pas le mêmes que celle du compilateur, qui sont différentes de celles du programmateur etc… **Si au cours de cette séance vous ne trouvez pas le menu indiqué sur le texte du sujet, vérifiez que vous êtes bien dans la bonne fenêtre.**

2. Présentation de la carte d'évaluation

Elle contient un CPLD de la série MAX 7000S, le circuit EPM7128SLC et un hybride CPLD/FPGA de la série FLEX10K, le circuit EPF10K20. Des cavaliers sur la carte permettent la programmation de l'un ou l'autre ou des deux circuits.
Lors de cette séance nous utiliserons le circuit EPM7128SLC. Vérifier dès maintenant que les cavaliers sont positionnés comme le précise la documentation de la carte (voir aussi en annexe). Ceux-ci se trouvent entre le circuit EPM7128SLC et le connecteur JTAG_IN du câble de programmation ; dans notre cas, ils doivent être tous les 4 en position haute (voir en annexe).
Le circuit EPM7128SLC est associé à un oscillateur de 25,175 MHz (borne 83) et à deux afficheurs sept segments (voir documentation ou annexe pour les bornes de sorties), les sorties étant actives au

NL0. Il est également possible de connecter des diodes électroluminescentes et de boutons poussoirs à n'importe quelle entrée sortie du circuit par câblage d'un fil (hook-up wire).
La carte doit être alimentée par une tension continue de 7 à 12 V. Elle doit être de plus connectée depuis son connecteur JTAG_IN au port parallèle de l'ordinateur par le câble "ByteBlasterMV".

3. Installation du logiciel

Si le logiciel est déjà installé, aller directement à la fin de ce paragraphe pour vérifier que la licence valide bien toutes les options.

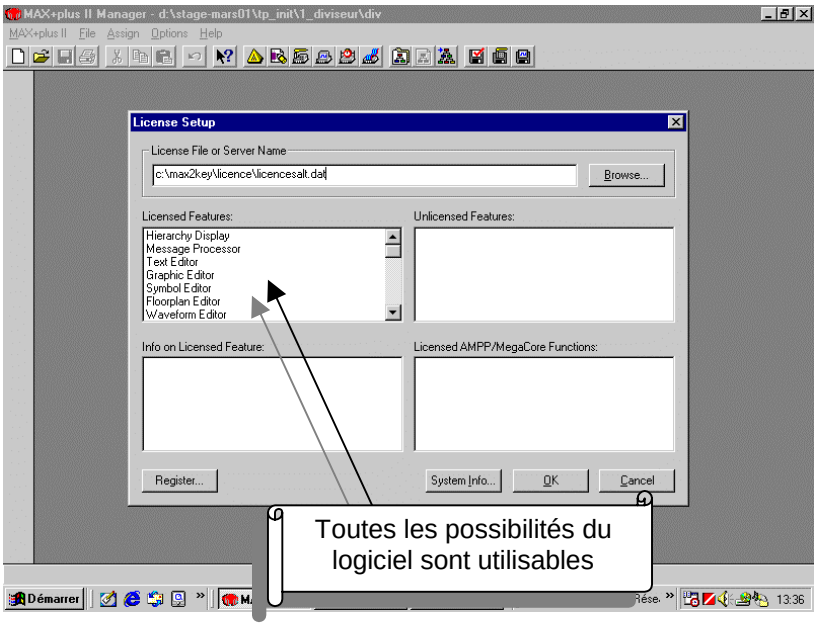
Pour installer Max+plus, lancer Windows, mettre le CDROM dans le lecteur et suivre les instructions pour une installation complète (**Full installation**).

Créer un raccourci (clic droit sur le bureau puis **nouveau / raccourci / parcourir**) vers le fichier **c:\maxplus2\maxstart.exe**.

Installer la clé matérielle sur le port parallèle de l'ordinateur ainsi que le câble **ByteBlaster** et la carte d'évaluation.

Lancer le logiciel (double clic sur le raccourci créé).

Installer le fichier licence. Celui-ci est un fichier texte récupéré par e-mail auprès du serveur Altera moyennant la fourniture d'un numéro d'identification ainsi que du numéro de la clé matérielle.
Il faut ensuite changer l'extension pour obtenir un fichier **.dat**.
Si l'installation vient d'être effectué, ce dernier se trouve actuellement sur la disquette ou le CDROM fournis, sous le nom "**licence.dat**". Le copier dans le répertoire **c:\max2key** (ou tout autre au choix).
Déclarer ensuite le chemin dans le logiciel : **Option / Licence setup / Browse**, sélectionner par un double clic "**licence.dat**" pour qu'il s'écrive dans "**file name**" ; valider par **OK**.
Vérifier que toutes les options sont bien validées : **Option / Licence setup**, la fenêtre suivante doit alors apparaître, où toutes les caractéristiques sont dans la partie «licenciée» :



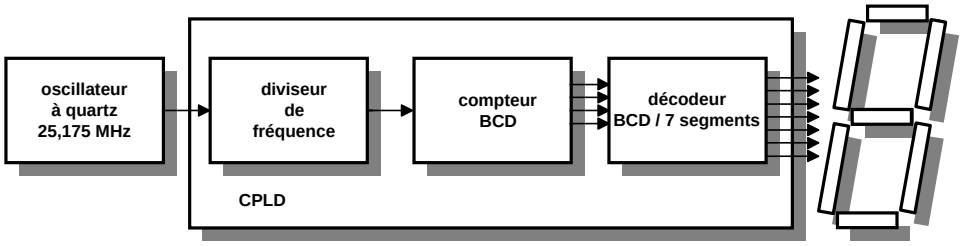
Remarque : La firme Altera propose également sur son site internet une version étudiante gratuite de Max+plus II. La principale différence avec la version professionnelle est l'impossibilité de programmer les circuits les plus rapides (ceux de notre carte de développement n'en font pas partie). Après

quelques versions laborieuses, les dernières déclinaisons (version 10.0) semblent fonctionner correctement. Pour obtenir la licence par mail, il suffit alors de donner le n° du disque dur du PC sur lequel le logiciel sera installé (détail sur le site http://www.altera.com).

Déclarer la carte d'évaluation : **Max+plus II \ Programmer** puis **Options \ Hardware Setup** puis déclarer le câble **ByteBlaster** et le port **LPT1** dans les boîtes de dialogue appropriées.

4. Présentation du projet

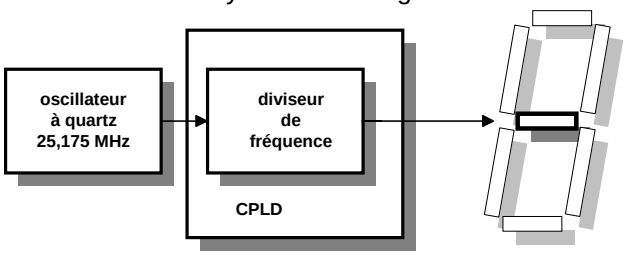
Comme nous l'avons précisé en introduction, nous allons implanter au sein d'un des circuits de la carte de développement, une fonction permettant, à partir de l'oscillateur à quartz de la carte, de visualiser l'écoulement secondes sur un des afficheurs. Le schéma synoptique de l'ensemble est le suivant :



Dans un premier temps, nous nous contenterons d'implanter le diviseur, qui nous permettra à partir du signal à 25,175 MHz d'obtenir un signal à 1 Hz. Nous allumerons alors une seule diode, toutes les secondes.
Nous effectuerons ensuite une description et une simulation du compteur, qui sera binaire dans un premier temps, puis binaire codé décimal (BCD) par la suite.
Enfin nous synthétiserons le projet complet.

5. Génération d'un signal à 1 Hz

Commençons donc par générer un signal de période 1 Hz à partir de l'horloge à 25,175 MHz. Pour cela il faudra réaliser un diviseur de fréquence. Nous ferons ensuite clignoter la DEL centrale du premier afficheur de la carte d'évaluation au rythme de ce signal.



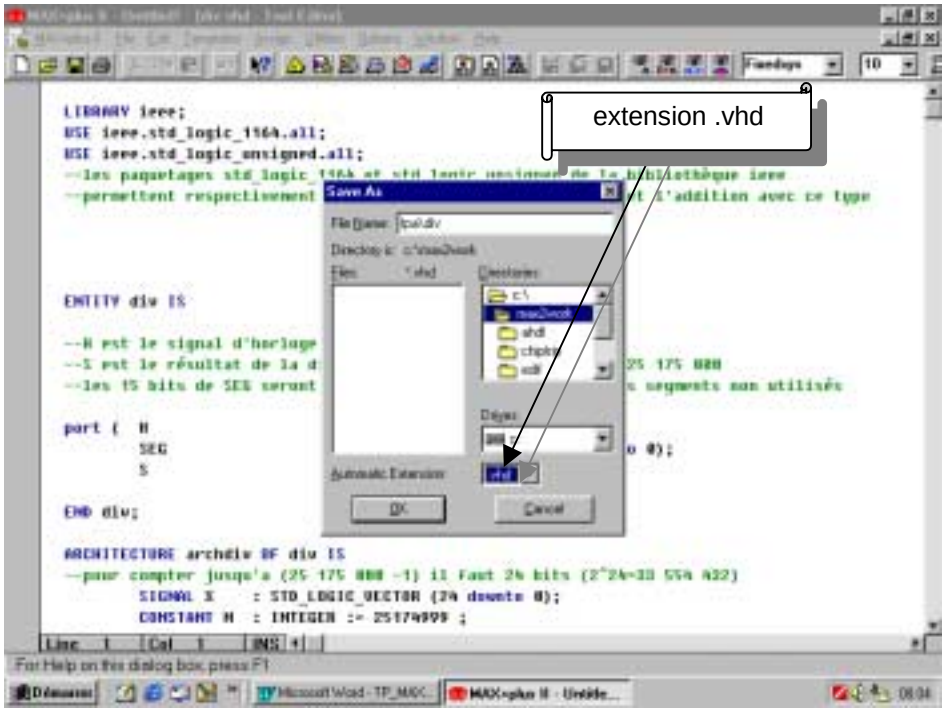
Au cours de ces opérations, nous utiliserons les fonctions suivantes du logiciel :
entrée d'un texte en VHDL,
choix du circuit cible et assignation des bornes du circuit aux entrées et sorties de notre projet,
compilation,
programmation du circuit

5.1. Description du diviseur de fréquence, assignation et compilation

5.1.1. Description

Les opérations d'écriture d'un fichier VHDL étant souvent laborieuses, surtout au début, on se servira du fichier div.vhd qui se trouve dans le répertoire **c:\max2work\tp_init\div** (ou éventuellement sur une disquette ou CDROM fourni).
Une version « papier » de cette description se trouve en annexe 1.

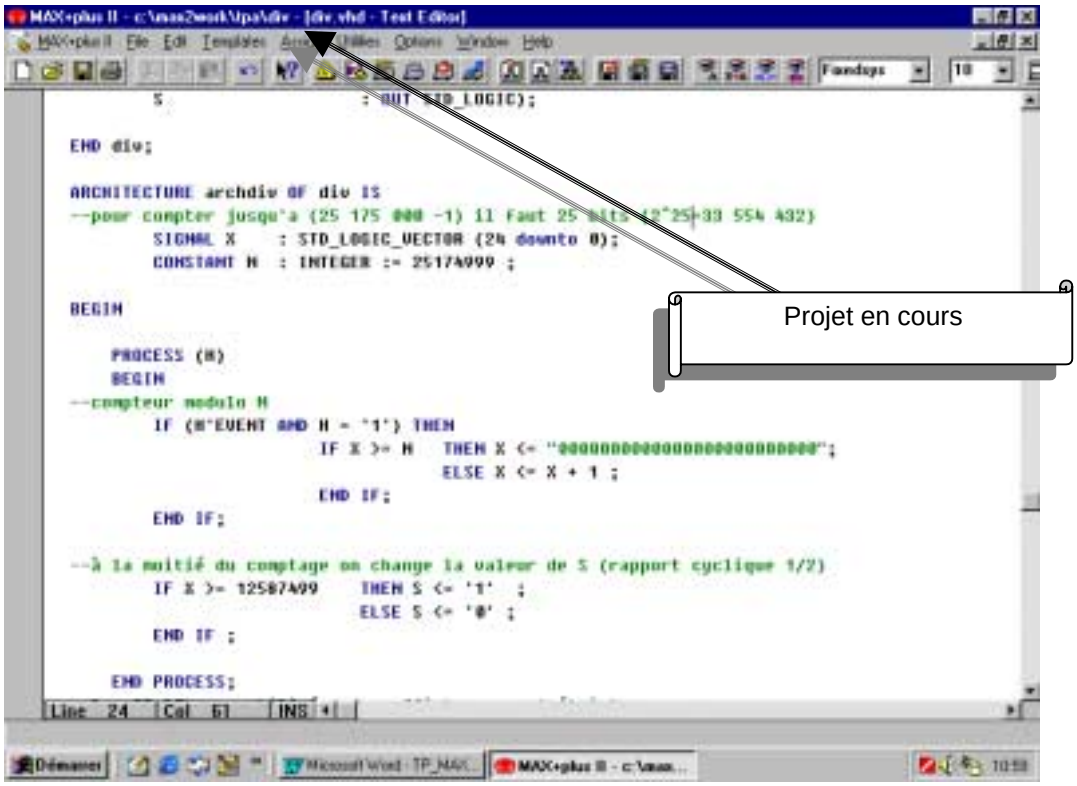
Ouvrir ce fichier (**File / Open)** puis le sauvegarder immédiatement (**File / Save As**) dans un sous-répertoire **tpa** du répertoire **max2work**. On disposera ainsi toujours de la version originale en cas de modification hasardeuse. Si le répertoire **tpa** n'existe pas, il sera créé par le logiciel. On n'oubliera pas d'autre part de sauvegarder le fichier avec l'extension « .**vhd** » afin qu'il soit reconnu comme un fichier VHDL.



Etudier le programme proposé : après déclaration des différentes entrée et sorties dans l'entité, l'architecture de la description permet de générer un signal S de rapport cyclique 0,5 et de fréquence 25 175 000 fois plus faible que le signal H. Les différents signaux SEG permettront d'éteindre les diodes que nous n'utiliserons pas sur les afficheurs (sans cette opération, ils éclairent de manière aléatoire). Les afficheurs de la carte étant à anode commune reliée à Vcc, l'extinction d'une diode se fait par imposition d'un niveau logique 1.

5.1.2. Compilation

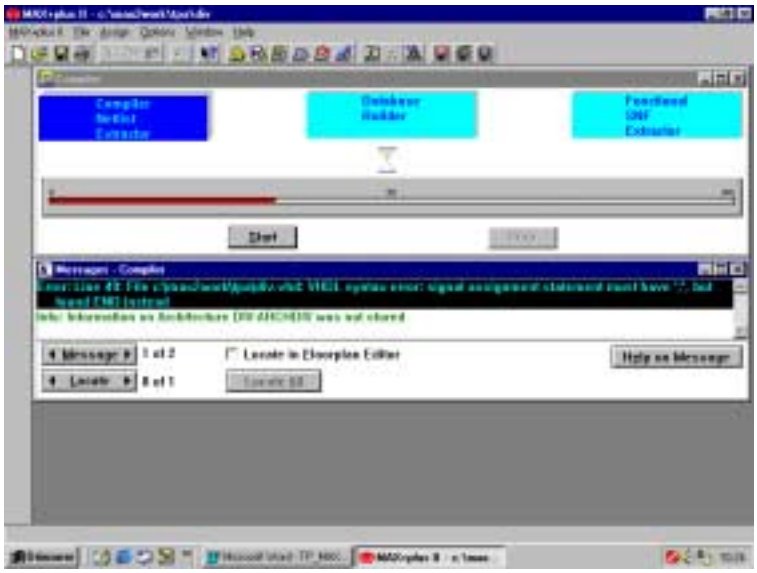
Vérifier que le fichier est bien le projet en cours (barre du haut) sinon le déclarer comme tel : **File / Project / Set Project to Current File**. On peut aussi utiliser **Ctrl Shift J**. Cette commande est à retenir car elle servira souvent. Pour éviter des erreurs, il est conseillé de l'exécuter avant chaque sauvegarde et compilation.



Compiler le projet : **File / Project / Save & Check** (ou bien **Ctr K**). L'un des deux écrans suivants (fonction du paramétrage du logiciel –voir plus loin-) apparaît alors :

Le compilateur ne signale normalement aucune erreur. Terminer la compilation en validant « **OK** » puis « **Start** ». Aucune erreur n'est signalée par le compilateur, qui génère simplement des avertissements, précisant les sorties du bus SEG restent systématiquement au niveau logique 1, ce qui est normal dans notre cas.

Pour visualiser ce qui se passe en cas d'erreur dans la description, revenir dans l'éditeur et générer volontairement une erreur en supprimant par exemple le point virgule à l'avant dernière ligne. Relancer la compilation.

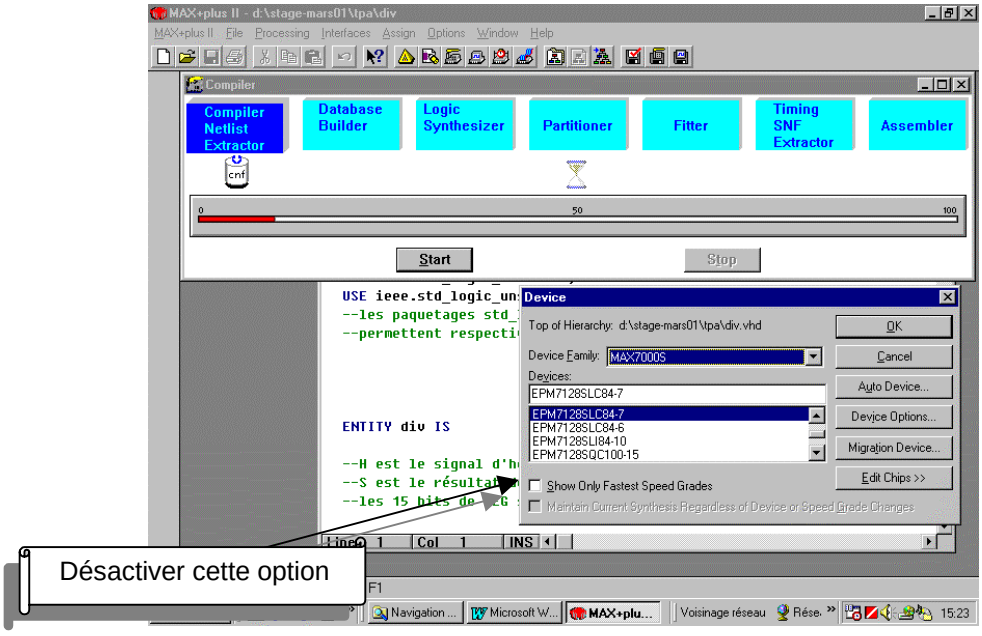


Le compilateur signale alors, « qu'après la déclaration d'assignement du signal, il trouve le mot END, alors qu'il attend un point virgule ».
Localiser cette erreur dans la description par **Message**, puis **Locate**.
Effectuer la correction et vérifier en recompilant que l'erreur a été corrigée.

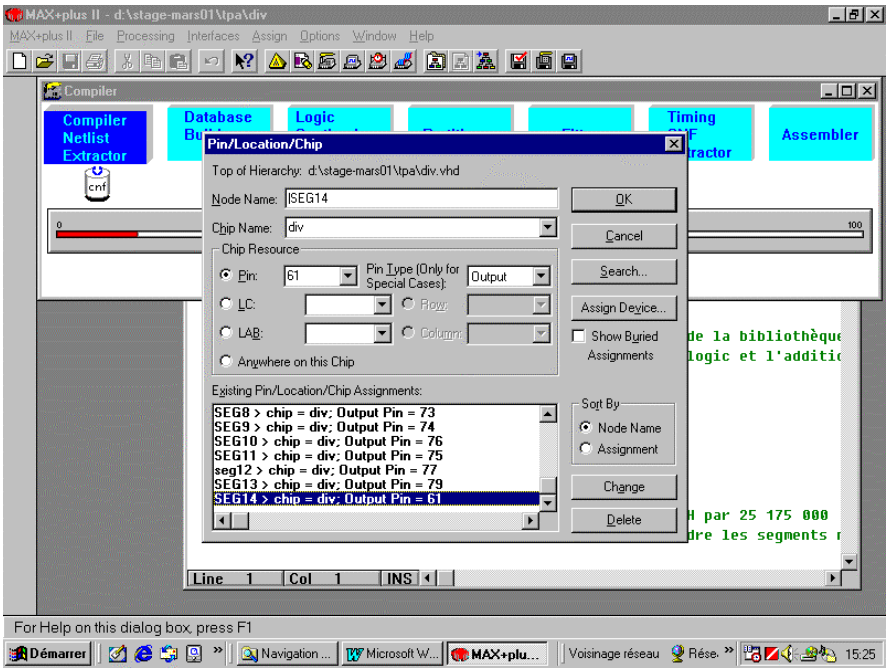
5.1.3. Assignation du circuit cible et de broches d'entrées sorties

Lors de l'opération que nous venons d'effectuer, nous n'avons pas précisé au compilateur le circuit cible que nous souhaitons utiliser, ni les broches de ce circuit que nous souhaitions affecter à nos entrées et sorties. Nous allons donc reprendre les étapes précédentes, mais en ajoutant cette fois des directives supplémentaires.

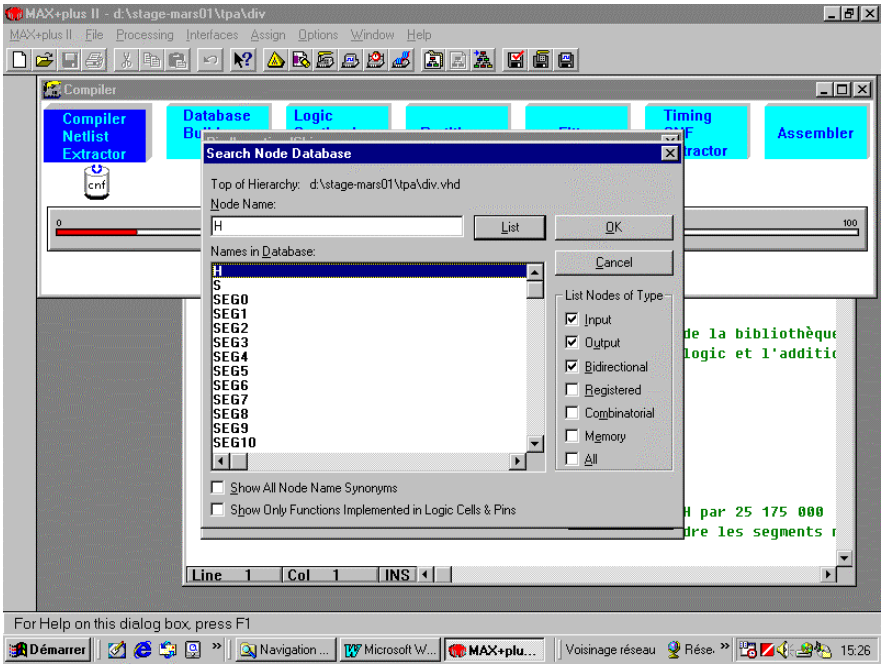
Ouvrir de nouveau le compilateur (**Ctrl K**).
Choisir le circuit cible : **Assign / Device** puis **MAX7000S** pour **Device Family**, désactiver l'option permettant la visualisation uniquement des composants les plus rapides et sélectionner le circuit **EPM7128SLC84-7** pour **Device**.



Affectons maintenant les broches externes du circuit aux différentes entrées/sorties de la description :
l'entrée H doit correspondre à l'oscillateur de la carte (borne 83)
la sortie S à la DEL choisie (borne 67)
les autres DELs des afficheurs doivent être éteintes par le bus SEG (bornes 58, 60, 61, 63, 64, 65, 68, 69, 70, 73, 74, 76, 75, 77 et 79).
Pour affecter une borne : **Assign / Pin / Location / Chip**.
On peut ensuite :
- soit donner le nom de l'entrée/sortie dans **Node Name** et le numéro de broche dans **Pin** puis choisir **Add** à chaque broche et **OK** uniquement lorsque toutes les affectations sont assignées :

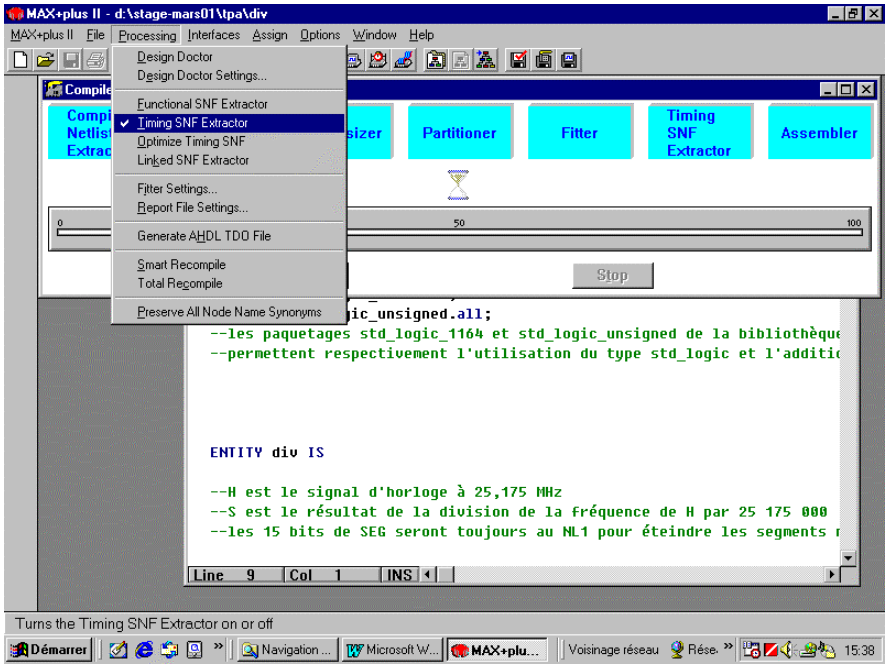


- soit utiliser l'option **Search**, puis **List**, puis **OK** pour chaque entrée/sortie.



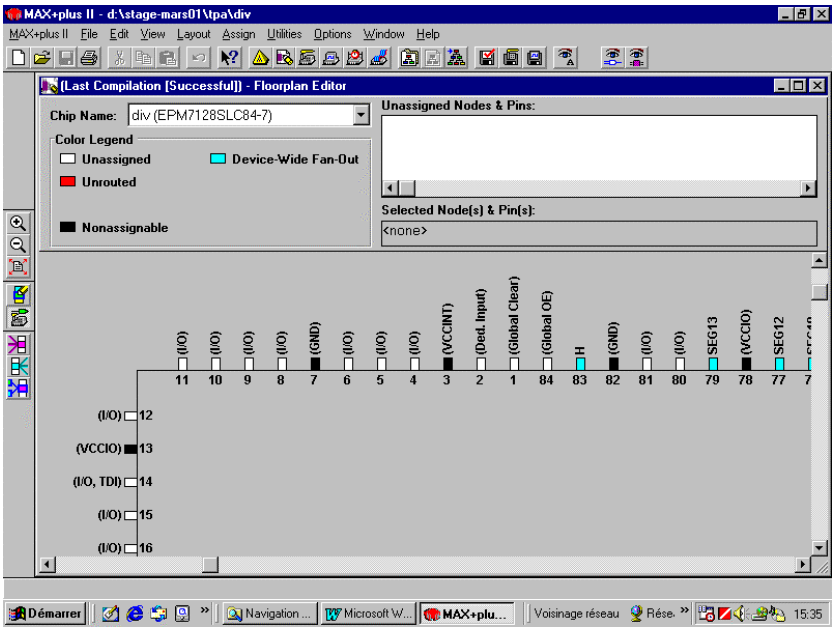
L'opération étant fastidieuse, il est conseillé de jongler entre les deux possibilités.

Vérifier que l'option **Timing SNF** est bien validée dans le menu **Processing** afin de permettre le placement et le routage des blocs logiques, sans quoi la programmation du circuit cible ne sera pas possible.



Continuer la compilation par **Start**. Le compilateur ne signale normalement aucune erreur, mais «s'étonne» de nouveau que certaines sorties restent au NL1. Il précise d'autre part que pour un meilleur placement des blocs logiques à l'intérieur du circuit cible, il est préférable de lui laisser faire l'assignation des broches (dans notre cas, la maquette étant figée, nous n'avons pas le choix).

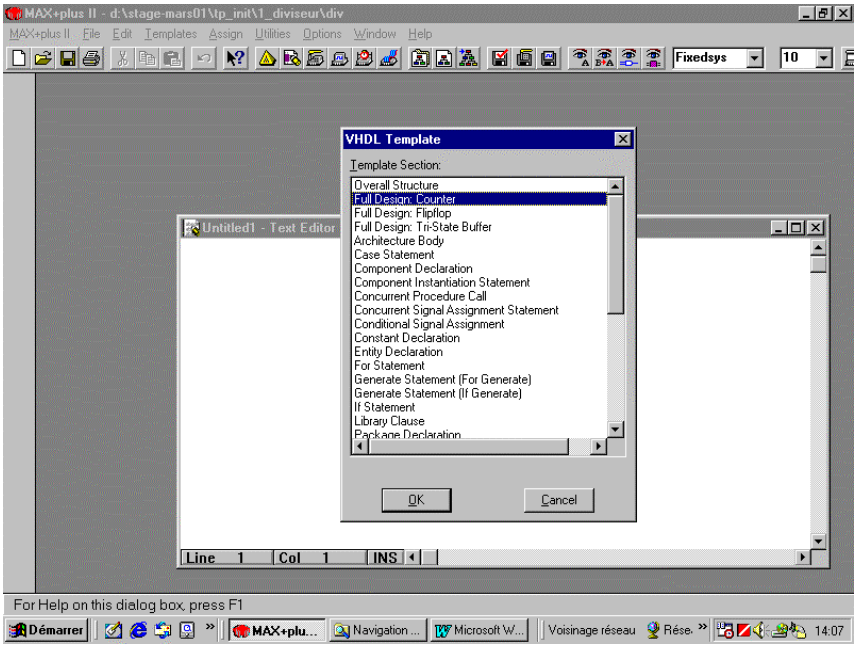
On peut vérifier l'affectation des broches dans l'éditeur d'implantation par **Max+plus II / Floorplan Editor** puis **Layout / Device View.**



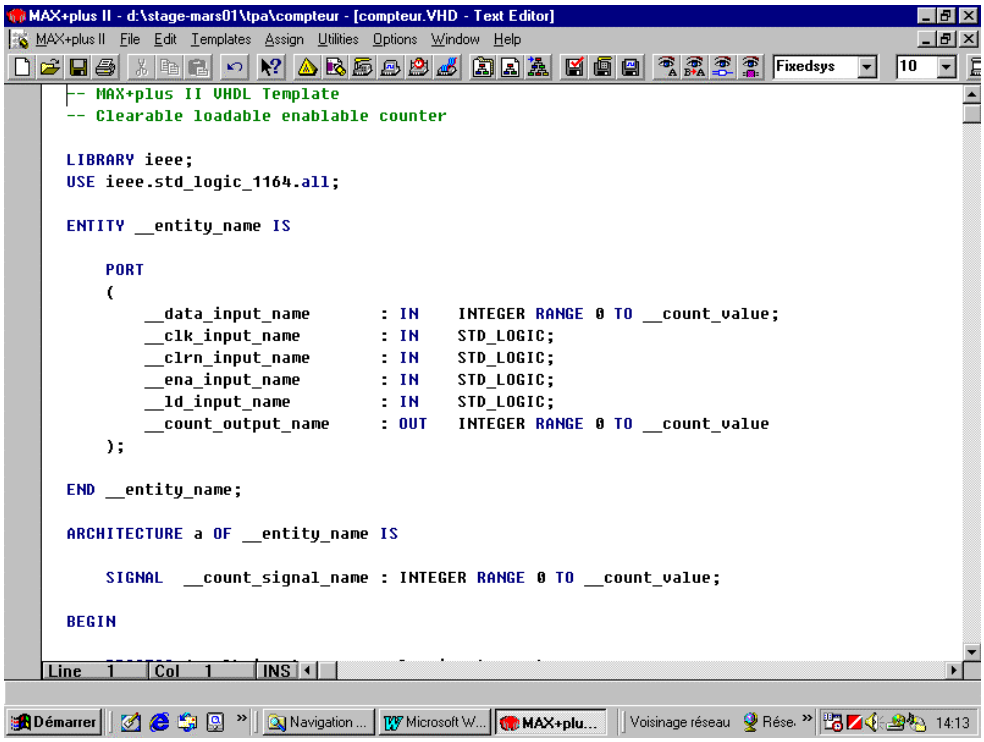
Fermer le compilateur et l'éditeur d'implantation.

5.1.4. Fonctions prédéfinies

Remarque : le logiciel propose des structures de descriptions VHDL prédéfinies. Pour y accéder : **File / New / Text Editor file** puis **OK** puis **Templates / VHDL Templates**, choisir par exemple **Full Design Counter** :

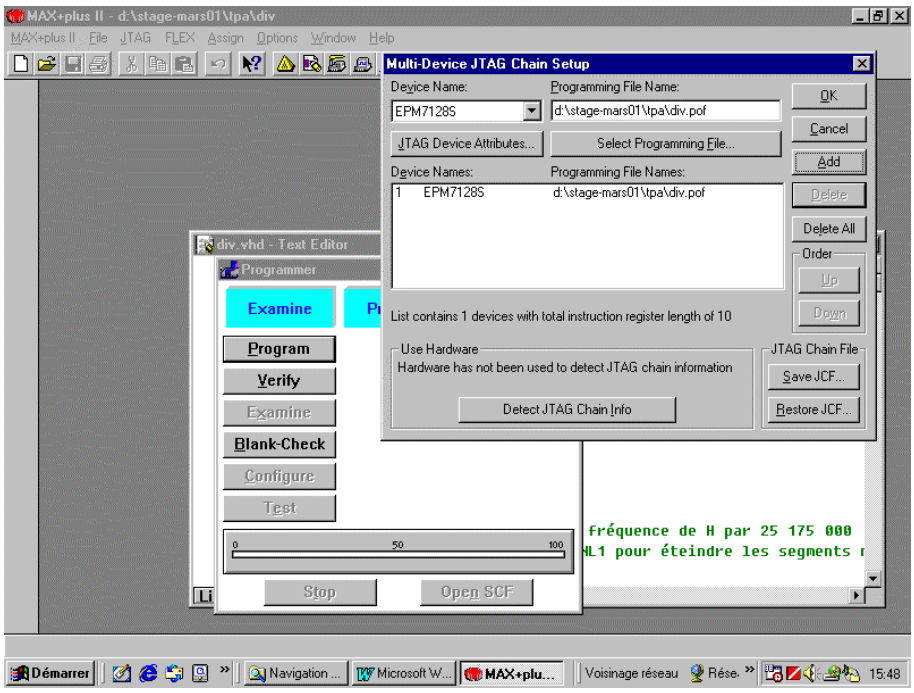


Le fichier ainsi ouvert propose une description d'un compteur en VHDL, où les noms des divers paramètres sont précédés de « __ » (ce qui générera une erreur à la compilation). Il suffit alors à l'utilisateur de remplacer ce nom par celui qu'il souhaite.



5.2. Programmation du circuit cible

Préparer la programmation du circuit. Pour cela appeler le programmateur par **Max+plus II / Programmer** ; puis déclarer la cible et le fichier de programmation : **JTAG / Multi-Device JTAG Chain Setup**.
Supprimer éventuellement toute déclaration précédente (**Delete All**) puis déclarer le nom du circuit cible dans **Device Name** et celui du fichier de programmation (**div.pof**, -pour program output file- on pourra utiliser le "butineur" **Select Programming File**...) dans **Select Programming File Name**.



Choisir **Add** puis vérifier la cohérence des informations que vous venez d'entrer avec celle détectée par le programmateur en cliquant sur **Detect JTAG Chain Info** puis **OK**.
Programmer le circuit : **Program** et vérifier le bon fonctionnement sur la carte.

Sortir du logiciel : **File / Exit Max+plus II**, éteindre l'alimentation de la carte, puis rallumer. Vérifier que la carte fonctionne toujours en justifiant.

6. Affichage des secondes

En cas de problème, les fichiers nécessaires aux exercices suivants se trouvent dans le répertoire **c:\max2work\tp_init\1_diviseur** (ou bien sur une disquette ou CDROM). On pourra éventuellement transférer les fichiers vers le répertoire de travail **c:\max2work\tpb** que l'on créera.

Nous allons maintenant commander l'affichage des secondes sur l'afficheur des unités au rythme de l'horloge S créée précédemment. Cette horloge incrémentera un compteur dont les sorties attaqueront un décodeur BCD - 7 segments.

La méthode utilisée dans la suite pour mener à bien le projet n'est pas optimale (découpage en sous-ensembles trop simples, mise bout à bout de sous-ensembles sans vision globale au départ…), notre objectif étant ici d'étudier les possibilités du logiciel.

Au cours de ces opérations, nous apprendrons à :

faire une description par chronogrammes d'un projet,
faire une simulation fonctionnelle puis temporelle,
utiliser les fonctions prédéfinis dans le logiciel
décrire un projet graphique général et le modifier après constatation d'une erreur (choix d'un compteur binaire au lieu de BCD).

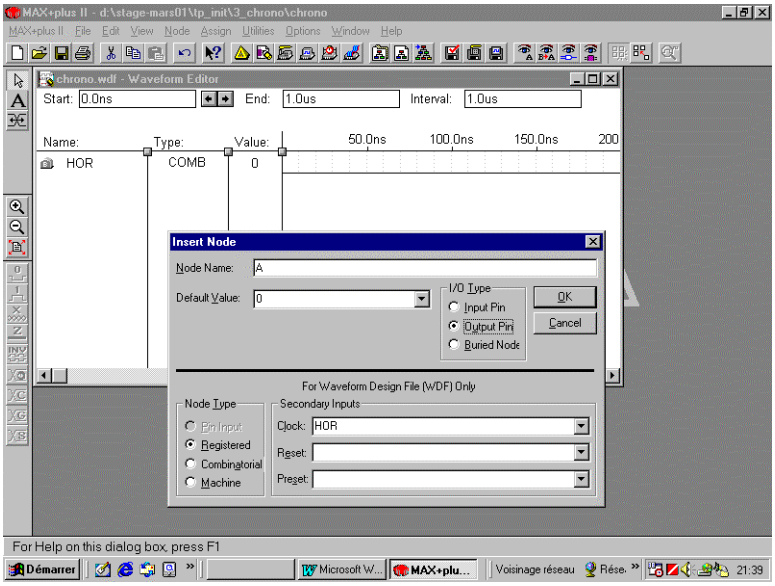
6.1. Description d'un compteur

Nous utiliserons l'éditeur de chronogramme pour décrire le compteur, qui sera binaire (et non BCD) par souci de simplification dans un premier temps.

Pour ouvrir l'éditeur : **File / New / Waveform Editor File** en **.wdf** (pour waveform desing file).

Sauvegarder dans le répertoire **tpb** et déclarer comme projet en cours.

Nous avons besoins d'une entrée d'horloge HOR, et des 4 sorties binaires A, B, C et D (du poids faible vers le poids fort). Pour placer ces entrées/sorties : **Node / Insert Node** (ou clic droit sur la feuille et **Insert Node**).



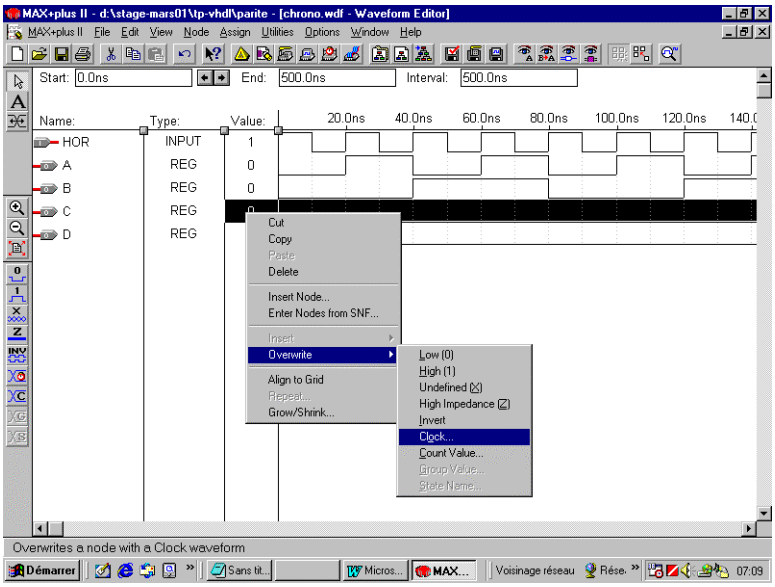
Puis préciser pour chaque nœud le nom, la valeur par défaut (à t=0), le type (entrée ou sortie, le terme **Buried Node** désignant un nœud interne au circuit) ; dans le cas d'une sortie préciser si elle est combinatoire ou séquentielle. Dans notre exemple les sorties sont séquentielles et on précisera que l'horloge de chacune est HOR (on n'utilise pas les entrées de mise au NL0 et NL1 des bascules).

Pour représenter un signal carré sur HOR, il faut fixer la période qui a peu d'importance ici (nous ne sommes pas obligés de respecter la période du signal S de une seconde). Cette période est fonction de la valeur de l'espacement de la grille : **Option / Grid Size**, choisir **10 ns** par exemple, la période minimale d'un signal sera alors de 20 ns.

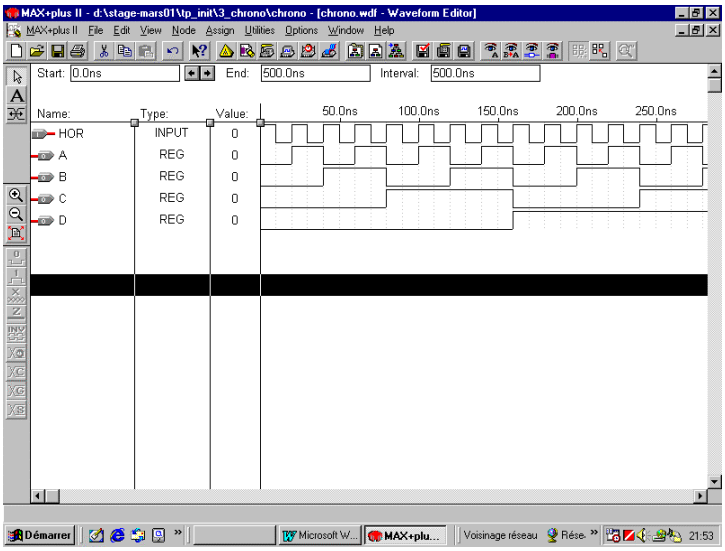
Il faut ensuite fixer une durée du chronogramme suffisamment longue pour pouvoir étudier tous les cas de figure, soit 16 fois 20 ns au minimum dans notre exemple ; prenons 500 ns. Pour cela : **File / End Time** puis **500 ns**.

Tracer ensuite l'horloge sur HOR : clic droit dans la colonne "**type**" de HOR puis **Overwrite / Clock** et choisir une multiplication par 1.

Notre compteur étant binaire A aura une période double de HOR, B une période quadruple etc. Tracer les quatre sorties par la même méthode que pour HOR en modifiant simplement le rapport de multiplication.



Le logiciel est prévu pour des sorties synchrones basculant sur le front montant de l'horloge ; pour cela complémenter cette dernière : clic droit dans la colonne "**type**" de HOR puis **Overwrite / Invert**.

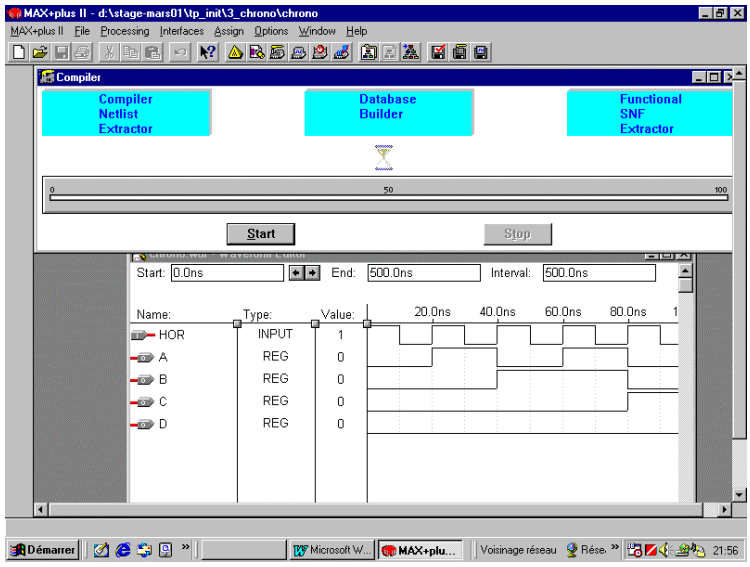


Au moment de la compilation, le compilateur génère les fichiers servant à la simulation. Suivant le type de simulation souhaitée (logique ou temporelle), il faut configurer le compilateur.

6.1.1. Compilation et simulation logique

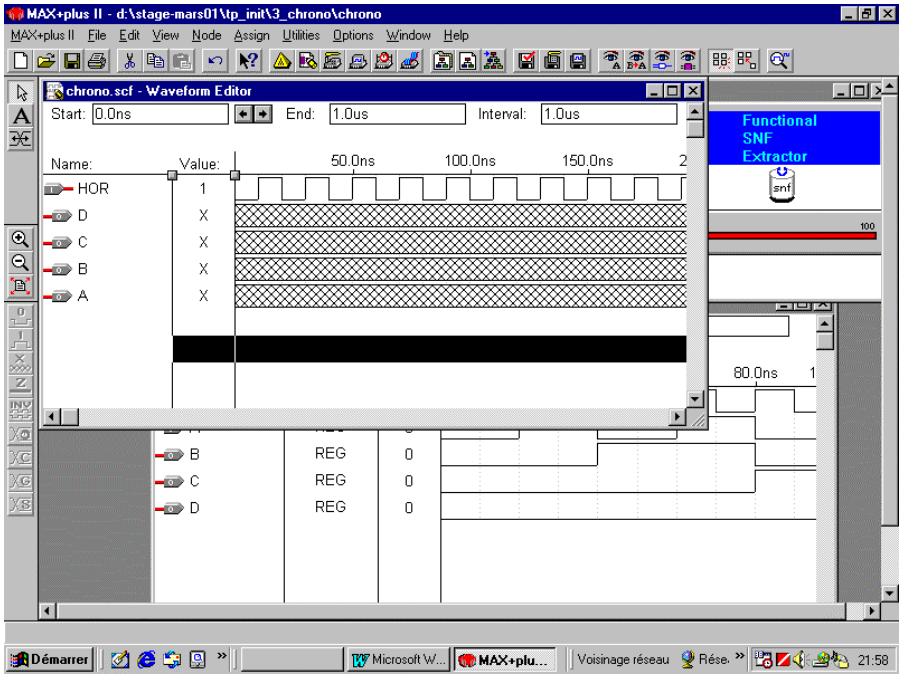
Ouvrir le compilateur : **Max+plus II / Compiler**.

Configurer ce dernier pour une simulation logique : valider dans le menu "**Processing**" la ligne "**Fonctional SNF Extractor**" (SNF pour Simulator Netlist File).

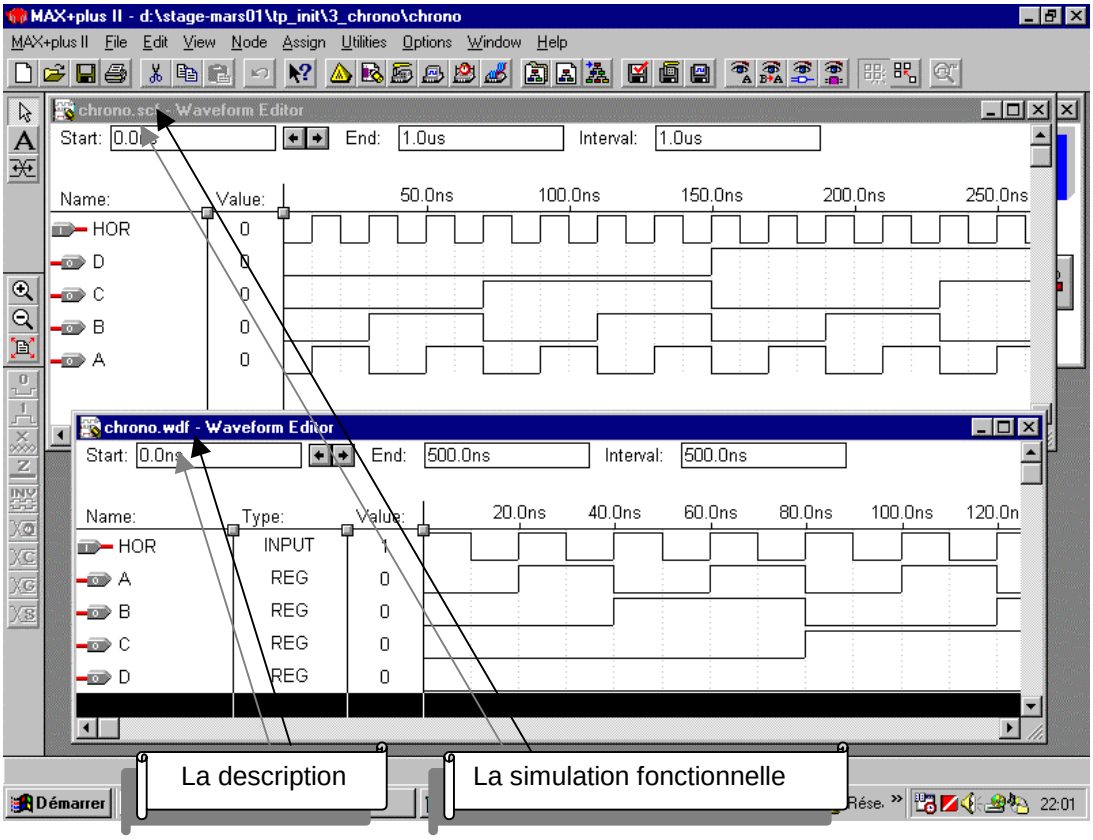


Lancer la compilation.

Pour la simulation, il faut générer un chronogramme décrivant l'évolution des entrées ainsi que les sorties que l'on souhaite visualiser. Ce chronogramme se crée comme pour la description du compteur par l'ouverture d'un fichier **Waveform Editor File** mais en ".**scf**" (pour simulator channel file). Ce dernier sera sauvegardé par le même nom que le fichier précédent (seule l'extension change).
Placer ensuite les nœuds par **Node / Enter Node from SNF** (on peut utiliser l'icône **List** pour sélectionner le nom des nœuds du projet en cours dans la boîte de dialogue de gauche puis les envoyer dans celle de droite).
Préciser ensuite l'évolution des entrées (ici HOR qui sera un signal carré de période indifférente, les critères de temps de propagation n'intervenant pas pour une simulation fonctionnelle).



Lancer la simulation : **Max+plus II / Simulator / Start**. Si tout c'est bien passé le simulateur ne signale aucune erreur et on peut ouvrir le fichier résultat de la simulation (**Open scf**). On constate alors que le résultat de la simulation est identique au fichier d'entrée :



Vérifier qu'il s'agit bien d'une simulation logique (temps de propagation nuls) et que le fonctionnement est bien celui escompté.

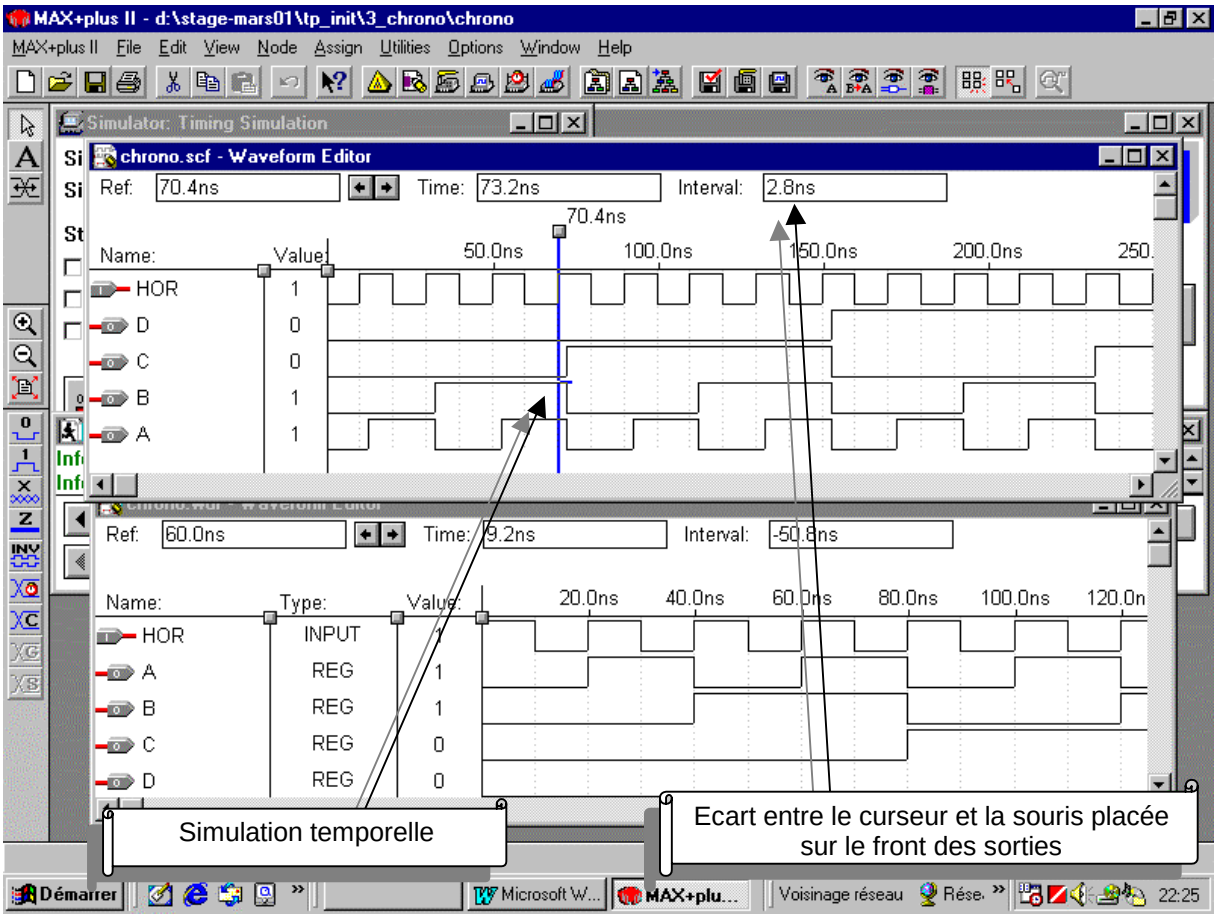
6.1.2. Compilation et simulation temporelle

Reprendre la compilation en inhibant "**Fonctional SNF Extractor**" dans **Processing**, ce qui a pour effet de valider "**Timing SNF Extractor**" dans le menu. Les temps de propagation dépendant du circuit cible choisi, on peut laisser le compilateur choisir dans la famille MAX7000S par exemple : **Assign / Device** puis **MAX7000S** pour **Device Family** et **Auto** pour **Device**.

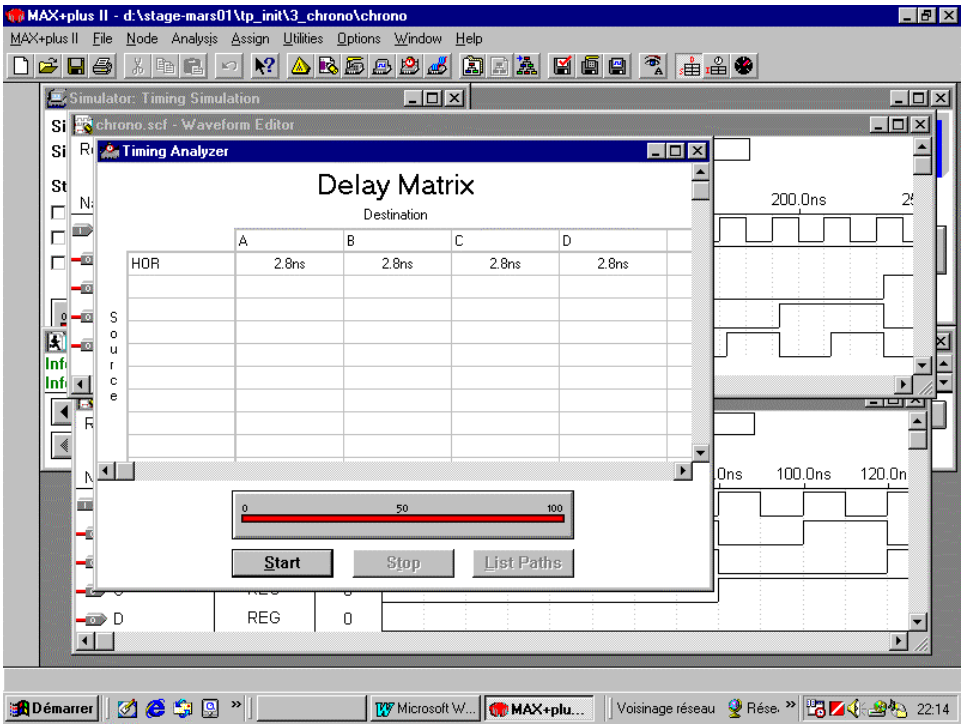
Lancer la compilation, puis la simulation (on utilise le même fichier ".scf" que précédemment dont les grandeurs de sorties seront recalculées par le simulateur).

Le compilateur précise son choix de circuit cible.

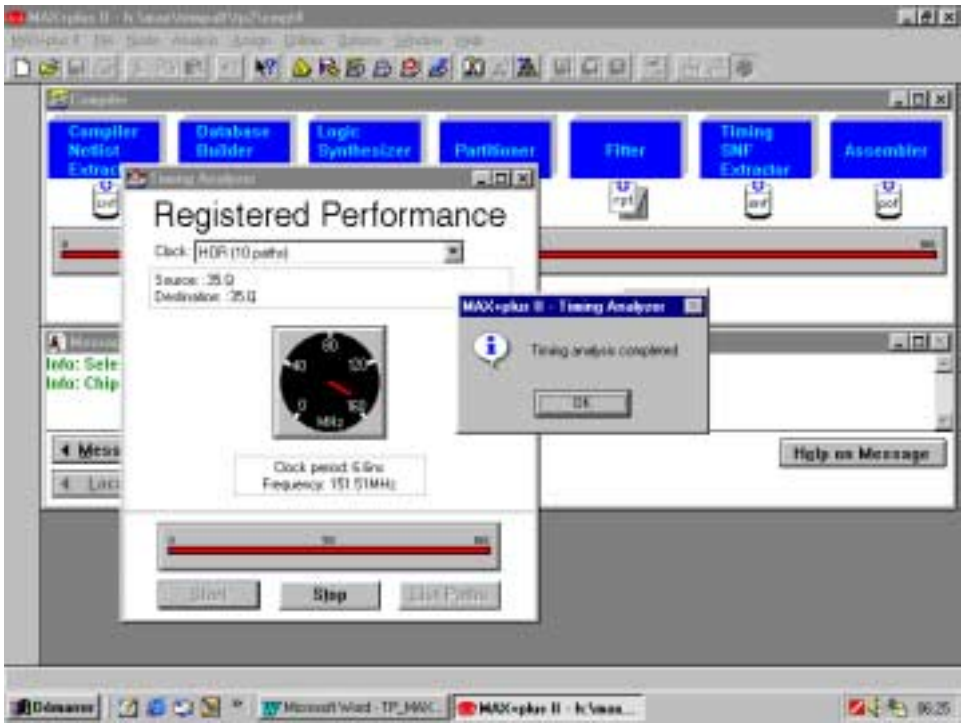
Noter dans la simulation l'influence des temps de propagation, d'autant plus visible que la fréquence de l'horloge est importante. Notre compteur est synchrone, toutes les sorties dans notre description du compteur avaient pour horloge le signal HOR. On peut relever un temps de propagation des bascules d'environ 2,8 ns. Pour cette lecture, après un zoom éventuel (à l'aide des loupes), on place le curseur (qui se crée en cliquant sur le chronogramme) sur un front d'horloge et la souris sur le front du signal de sortie ; l'intervalle de temps est alors affiché en haut à droite de la fenêtre. Il faut bien noter que cette valeur n'est pas la seule à prendre en compte pour le calcul de la fréquence maximale de fonctionnement.



Il est également possible de lire directement cette valeur dans l'analyseur temporelle : **Max+plus II / Timing Analyser**.



On peut également déterminer le vitesse d'horloge maximale de notre compteur, en modifiant le type d'analyse : dans la fenêtre de l'analyseur temporel, choisir **Registered Performance** du menu **Analysis**.
Le résultat dépend bien évidemment du circuit cible.



6.2. Description du projet général

A chaque fois qu'une description est compilée, un symbole graphique associé à la description est créé, ce symbole pouvant être utilisé dans une description graphique de niveau supérieur, ce que nous allons expérimenter maintenant.

La description du projet général d'affichage des secondes se fera par une description graphique assemblant les symboles :
- du diviseur décrit en VHDL dans la première partie du TP
- du compteur étudié précédemment
- d'un décodeur BCD - 7 segments de référence 7446 pris dans la bibliothèque se trouvant à **c:\maxplus2\max2lib\mf** fournie avec le logiciel.
- de symboles divers (entrées, sorties, alimentation…) pris dans la bibliothèque **c:\maxplus2\max2lib\prim**.

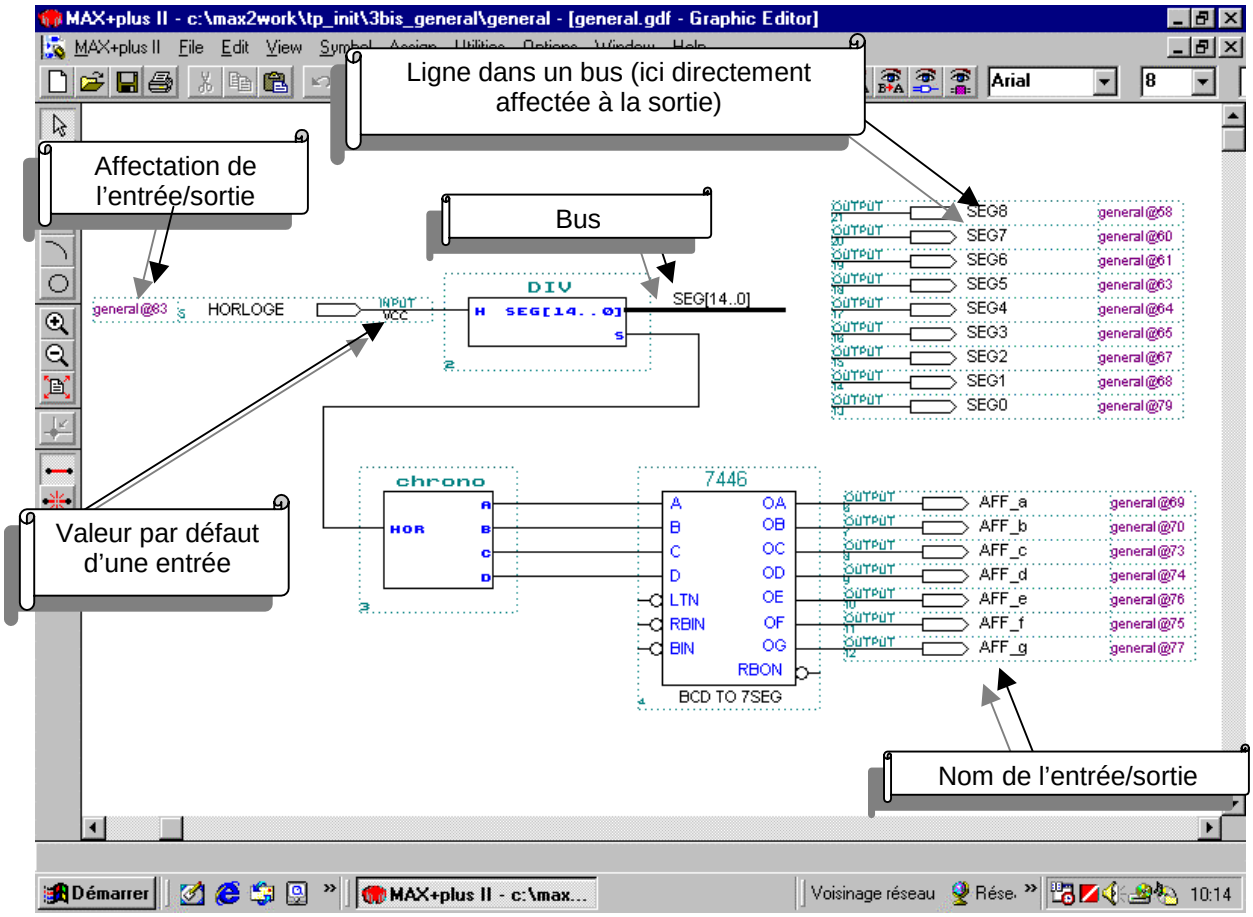
Ouvrir et sauvegarder un nouveau fichier graphique : **File / New / Graphic Editor file** (en "**.gdf**" pour graphic design file) puis **File / Save as** "**general.gdf**".

Nous allons utiliser pour ce projet des ressources (la description du diviseur et du compteur) qui sont peut être sauvegardées dans un autre répertoire. Si tel est le cas (hormis pour les répertoires **max2inc** et **max2lib**), il est nécessaire de déclarer au logiciel la localisation de ces ressources : **User Librairies** du menu **Option**.

Déclarer maintenant le fichier **general.gdf** comme projet courant.

Entre les symboles nécessaires : **Symbol / Enter Symbol**. Il est possible d'accéder au même menu par un clic droit.

Les relier au moyen des outils proposés sur le côté gauche de l'écran de manière à obtenir le schéma suivant :



Les DELs inutilisées des afficheurs sont placées au NL1 par le bus SEG de DIV. Cette solution (nous aurions pu aussi utiliser le symbole Vcc) nous permet de nous familiariser avec le tracé d'un bus : celui-ci diffère du tracé d'une ligne par l'utilisation d'un trait plein épais (clic droit puis **Line Style**). On notera également la manière de repérer un bus (exemple SEG[14..0]) et une ligne particulière dans un bus (exemple SEG8)

Les entrées non utilisées du 7446 sont reliées à Vcc par défaut, elles pourront donc être laissées en l'air. Pour voir le schéma interne du circuit, cliquer deux fois sur le symbole. Pour avoir accès à la documentation et la table de vérité, sélectionner l'icône « **?** » puis cliquer gauche sur le symbole du composant

Les noms des entrées/sorties sont écrits en sélectionnant puis en changeant le nom par défaut

Par un double clic gauche sur un symbole quelconque du schéma, on peut vérifier qu'il correspond bien à la description attendue (normalement) : une fenêtre s'ouvre alors sur cette description.

Sélectionner maintenant le circuit cible **EPM7128SLC84-7**.

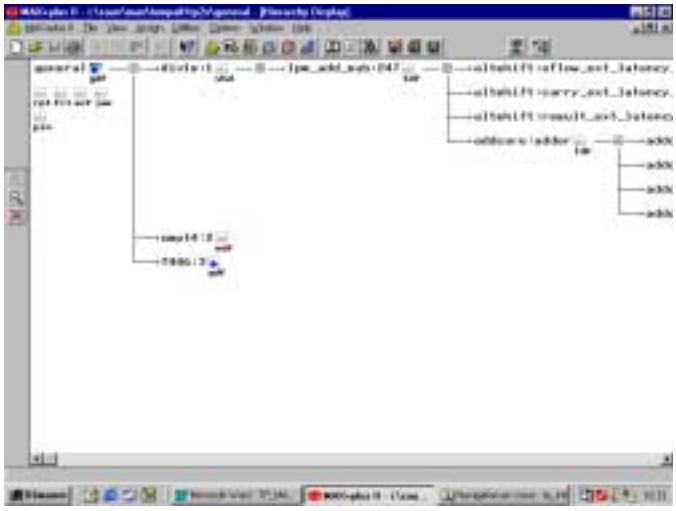
Assigner les broches du circuit cible pour être compatible avec la carte d'évaluation.(pour utiliser l'option **Search** le projet doit avoir été compilé auparavant)

Lancer une compilation avec analyse fonctionnelle dans un premier temps (en cas d'erreur, la compilation dure moins longtemps) puis avec une analyse temporelle par la suite (indispensable pour placer et router le circuit).

Programmer le circuit cible.

Observer le fonctionnement. Comme on pouvait s'y attendre, l'utilisation d'un compteur binaire au lieu d'un BCD produit des états indésirables.

Remarque : lorsque le projet imbrique un nombre important de sous-ensembles, il peut être intéressant de visualiser la hiérarchie existant entre les différentes descriptions ; cette fonctionnalité est accessible grâce à l'éditeur de hiérarchie : depuis l'éditeur graphique, **Max+plus II / Hierachy Display**.



6.3. Modification du projet

Ouvrir le fichier "**general.gdf**".

Double clic sur le symbole du compteur.
Sélectionner (avec précision à l'aide des marqueurs et des zooms) les états indésirables sur les 5 chronogrammes et les supprimer.
Attention : il faudra garder les états des sortie de 0 à 9, puis de nouveau l'état 0, faute de quoi le compteur synthétisé sera de nouveau binaire (c'est le plus simple à réaliser) si on laisse indéterminé l'état suivant 9.
Diminuer la durée des chronogrammes d'un temps équivalent à la partie supprimée (faute de quoi les entrées/sorties ne sont plus définies en fin de chronogrammes).

Déclarer comme projet en cours, sauvegarder, compiler puis fermer.

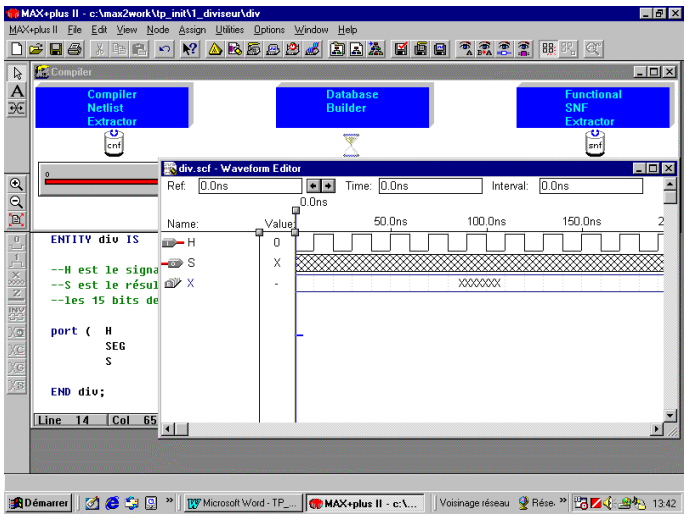
Déclarer le projet "general" comme projet en cours, sauvegarder, compiler et programmer le circuit.

Vérifier que la modification a bien été prise en compte.

7. Initialisation d'une simulation

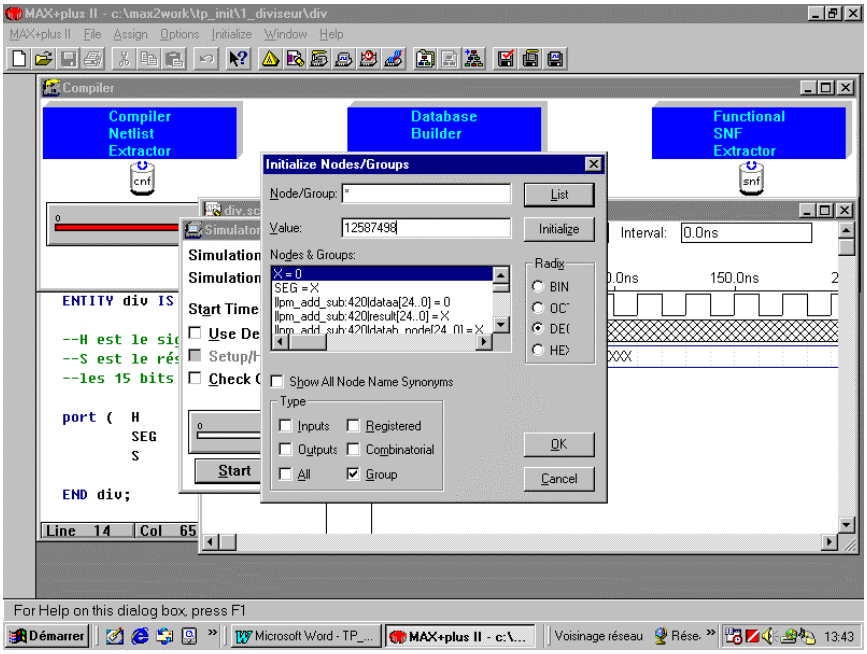
La simulation (fonctionnelle ou temporelle) d'un projet nécessite parfois d'imposer des conditions initiales au simulateur. Par exemple, pour visualiser le passage au NL1 de la sortie S du diviseur étudié dans la première partie, il faut attendre 12 587 499 coups d'horloge, ce qui nécessite un temps de calcul très long pour le simulateur.
Nous allons donc voir maintenant comment imposer ces conditions initiales au simulateur, en reprenant le cas du diviseur.

Ouvrir le fichier VHDL contenant la description, le compiler pour une simulation fonctionnelle (plus rapide).
Créer un fichier div.scf et placer le signal d'horloge d'entrée, le signal X de comptage ainsi que la sortie.
Imposer un signal carré pour H.



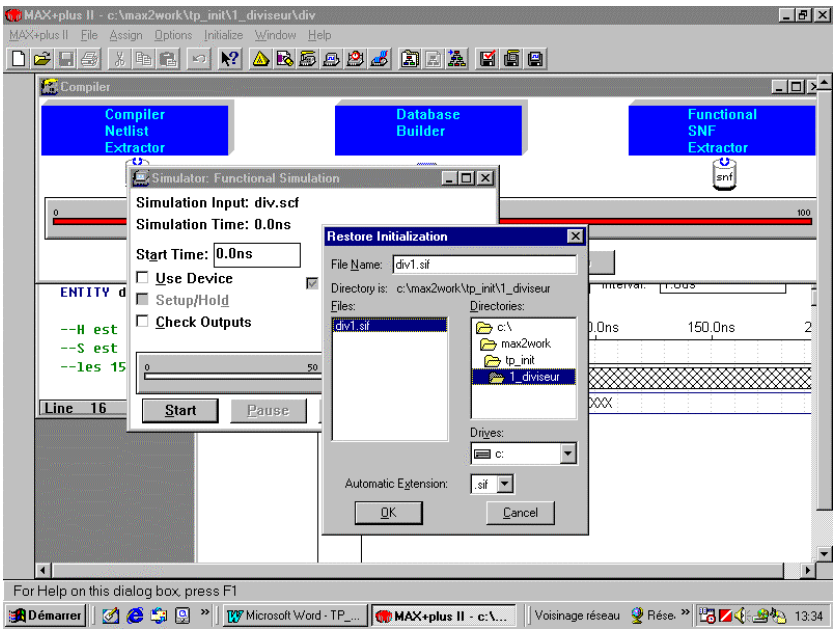
Lancer le simulateur (**Max+plus II / Simulator**)

Créer un fichier d'initialisation : **Initialize / Initialize Nodes / Groups**.
Afin de simplifier l'opération, plutôt que d'initialiser toutes les valeurs binaires une par une, nous allons travailler sur l'ensemble d'un bus.
Activer l'option **Group** uniquement, puis **List**.
Sélectionner X puis dans value imposer 12587498 si la représentation (**radix**) décimale est sélectionnée et C011EA pour une représentation hexadécimale, puis **Initialize** puis **OK**.

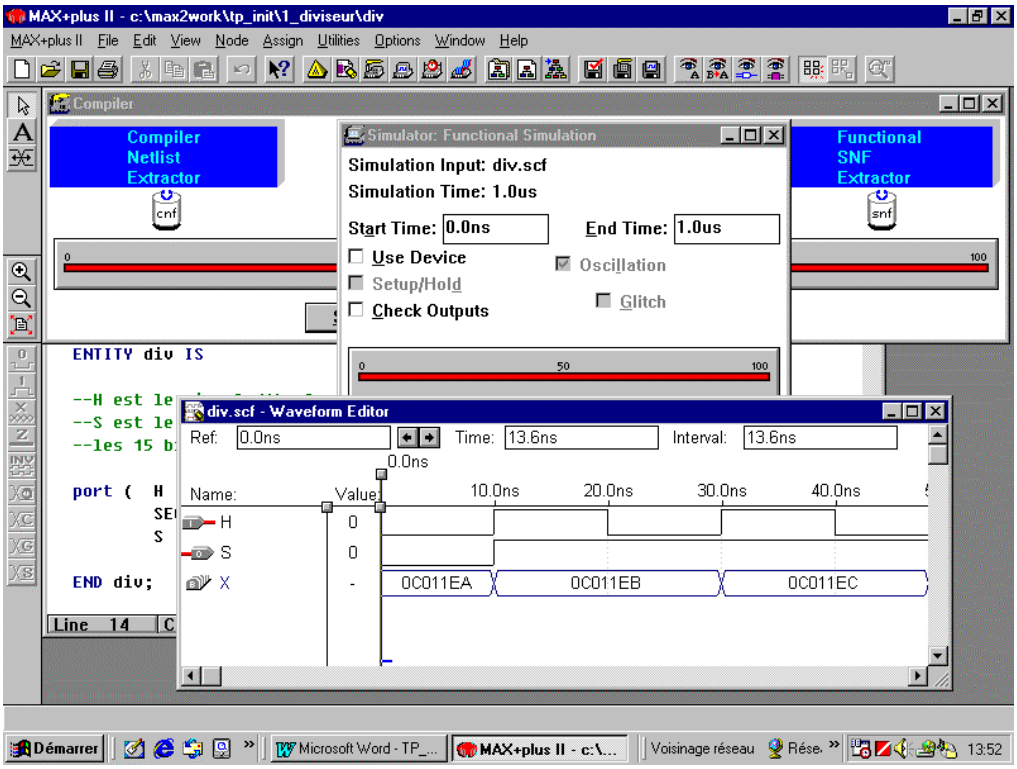


Sauvegarder l'initialisation : **Initialize / Save Initialization as** puis l'appeler div1.sif (sif pour simulator initilization file) par exemple. Cette sauvegarde devra être rappelée par

Initialize / Restore Initialization à chaque fois que l'on souhaitera démarrer la simulation dans les mêmes conditions.



Lancer la simulation, ouvrir le fichier scf, utiliser la loupe afin de voir les valeurs affichées (en hexadécimal) pour X.



Remarque : lorsqu'un signal (au sens VHDL) ou une variable interne est affectée à une sortie, c'est le signal ou la variable qu'il faut initialiser.

Annexe 1 : description VHDL du diviseur

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
--les paquetages std_logic_1164 et std_logic_unsigned de la bibliothèque ieee
--permettent respectivement l'utilisation du type std_logic et l'addition avec ce type


ENTITY div IS
--H est le signal d'horloge à 25,175 MHz
--S est le résultat de la division de la fréquence de H par 25 175 000
--les 15 bits de SEG seront toujours au NL1 pour éteindre les segments non utilisés

port (    H                 : IN STD_LOGIC;
          SEG               : OUT STD_LOGIC_VECTOR (14 downto 0);
          S                 : OUT STD_LOGIC);
END div;



ARCHITECTURE archdiv OF div IS

--pour compter jusqu'à (25 175 000 -1) il faut 25 bits (2^25=33 554 432)
          SIGNAL X        : STD_LOGIC_VECTOR (24 downto 0);
          CONSTANT N  : INTEGER := 25174999 ;

BEGIN

          PROCESS (H)
          BEGIN
--compteur modulo N
                    IF (H'EVENT AND H = '1') THEN
                                        IF X >= N         THEN X <= "000000000000000000000000";
-- l'affectation peut s'écrire plus simplement X<=(others=>'0')
                                                          ELSE X <= X + 1 ;
                                        END IF;
                    END IF;

--à la moitié du comptage on change la valeur de S (rapport cyclique 1/2)
                    IF X >= 12587499        THEN S <= '1'  ;
                                                ELSE S <= '0' ;
                    END IF ;

          END PROCESS ;

--les 15 DEL non utilisées des afficheurs sont éteintes
          SEG <="111111111111111" ;
-- ou plus simplement SEG<=(others =>'1')

END archdiv;
```

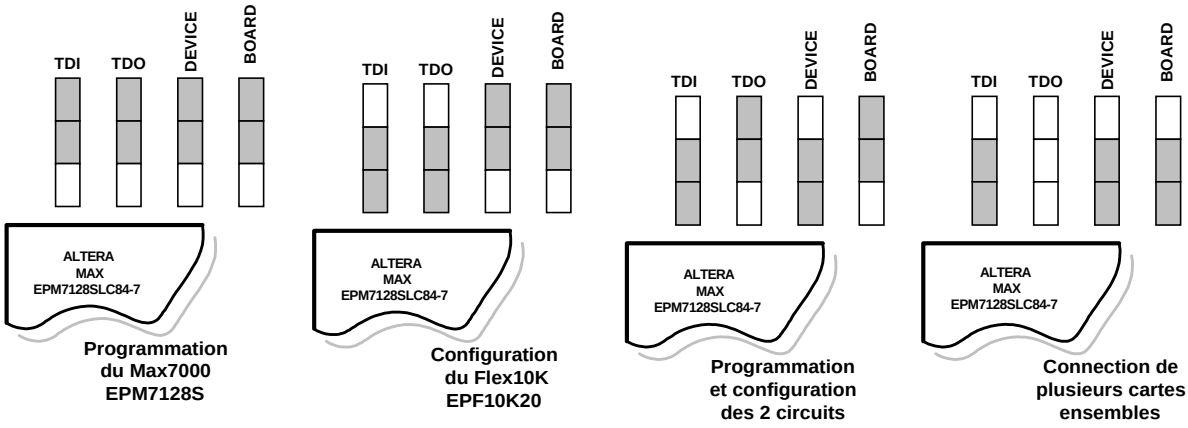
Annexe 2 : connectique de la carte de développement

Alimentation continue

Tension continue de 7 à 9 V, 250 mA minimum, polarité positive au centre du connecteur.

Cavaliers de configuration

Suivant l'utilisation souhaitée de la carte de développement, le cavaliers se trouvant au-dessus du circuit MAX7000 doivent être configurés comme indiqué ci-dessous (les zones grisées représentent les bornes en contact) :

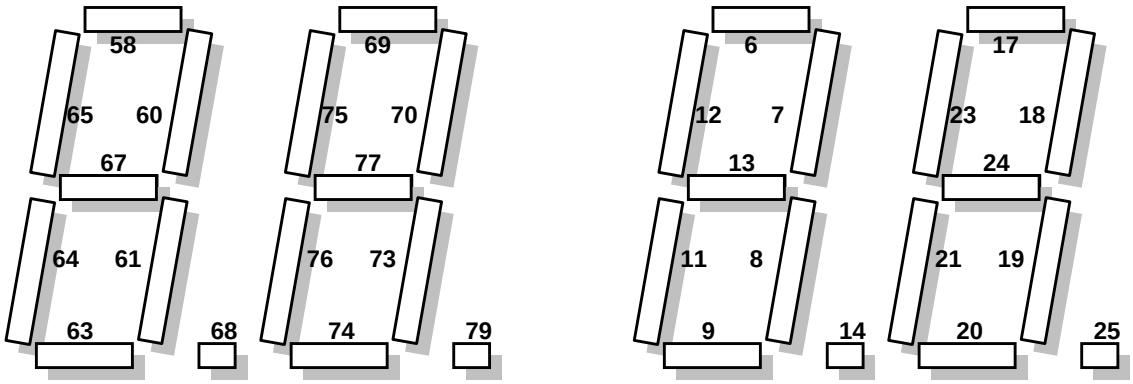


Oscillateur

Fréquence de 25,175 MHz, relié à la borne83 de l'EPM7128S et à la borne 91 de l'EPF10K20.

Afficheurs

Les deux groupes d'afficheurs sont reliés aux bornes des circuits comme indiqué ci-dessous :



Annexe 3 : problèmes classiques

Apparition de message d'erreur concernant la licence, ou la clé :

- vérifier que la clé est connectée au port parallèle de votre ordinateur (attention aux commutateurs de ports) ;
- vérifier que toutes les options sont bien validées : **Option / Licence setup** dans l'éditeur.

A la compilation d'un fichier VHDL un message d'erreur indique que le fichier doit contenir une entité de même nom :

- sauvegarder le fichier sous le même nom que l'entité ou changer le nom de l'entité.

L'éditeur VHDL affiche les mots réservés de la même couleur que les autres :

- vérifier que votre fichier est sauvegardé avec l'extension « .vhd » ;
- vérifier que **Syntax_Coloring** du menu **Option** est bien validée.

A la compilation d'un fichier VHDL, le compilateur semble ne reconnaître aucun terme :

- vérifier que votre fichier est sauvegardé avec l'extension « .vhd » ;

Certaines instructions propres au VHDL version 93, par exemple « xnor », semblent ne pas être reconnues par le compilateur :

- dans le compilateur, vérifier avec le menu **Interfaces / VHDL Netlist Reader Settings** que l'option **VHDL 93** est bien validée.

L'assignation de numéros de broches aux entrées sorties de la description n'est pas possible :

- le choix du circuit cible à été laissé au compilateur (l'option **auto** est validée dans la case **devices** du menu **Assign / Device** dans le compilateur.

Lors d'une description hiérarchique, un sous-ensemble n'a pas de symbole :

- vérifier si la description du sous-ensemble est bien dans le répertoire en cours et sinon vérifier que le répertoire où se trouve la description a bien été déclaré dans **User Librairies** du menu **Option** ;
- vérifier s'il existe bien un fichier **.sym** (pour symbole) avec le nom du sous-ensemble ; sinon le créer en ouvrant le fichier description du sous-ensemble (généralement **.gdf**, **.vhd** ou **.wdf**), puis refaire une compilation (ou encore dans l'éditeur **File / Créate Default Symbol**).

Dans une description hiérarchique, le logiciel précise ne pas pouvoir ouvrir la description correspondant à un sous ensemble :

- vérifier si la description globale a bien été sauvegardée dans le répertoire souhaité ;
- vérifier si la description du sous-ensemble est bien dans le répertoire en cours et sinon vérifier que le répertoire où se trouve la description a bien été déclaré dans **User Librairies** du menu **Option** ;

Dans une description hiérarchique, le logiciel détecte une désadaptation entre les entrées sorties d'un symbole et la description associée :

- il y a eu une confusion de noms lors des compilations des descriptions des deux sous-ensembles ; relire (attentivement cette fois) le premier point « stop » de ce tp ;supprimer avec l'explorateur windows tous les fichiers concernés, sauf les descriptions (généralement **.gdf**, **.vhd** ou **.wdf**) ; ouvrir ces descriptions, les sauvegarder sous des noms corrects et différents ; refaire les compilations.

Problèmes divers lors de la programmation :

- vérifier que la carte de développement est connectée et alimentée (une DEL verte allumée) ;
- vérifier que le câble **ByteBlaster** est correctement déclaré sur le bon port ;
- vérifier que les cavaliers de la carte sont placés correctement ;
- vérifier que le circuit cible choisi lors de la compilation est bien celui de la carte (toutes les lettres doivent correspondre) ;
- vérifier que celui imposé au programmateur dans : **JTAG / Multi-Device JTAG Chain Setup**. est bien **EPM7128S** ou **FLEX10K20** suivant la cible choisie sur la carte de développement ;
- vérifier que le fichier de programmation imposé dans : **JTAG / Multi-Device JTAG Chain Setup**. est bien un fichier **.pof** (programm object file) pour le circuit **EPM7128S** ou un fichier **.sof** (SRAM object file) pour le fichier **FLEX10K20**.

Annexe 4 : principales extensions de fichiers

Fichiers de description

description graphique : **.gdf**
description vhdl : **.vhd**
description par chronogrammes : **.wdf**

Fichiers de simulation

description par chronogrammes des entrées ; sorties attendues : **.scf**
initialisation des entrées : **.sif**

Fichiers de programmation

composants EEPROM : **.pof**
composants SRAM : **.sof**

Fichiers divers

rapport de compilation : **.rpt**
assignation des broches : **.acf**
symbole graphique d'une description : **.sym**

Transport d'un projet d'un PC vers un autre.

La plupart des fichiers sont recréés lors de la compilation ; il suffit simplement de transférer les fichiers de description (globale et sous-ensembles) **.gdf**, **.vhd**, et **.wdf**, les fichiers de simulation .**scf** et **.sif**, ainsi que le fichier **.acf** d'assignation des broches.