Travaux pratiques : programmations des CPLD et FPGA, notions avancées avec le logiciel Max+plus II

Après avoir, lors d'une séance précédente, étudié les principales fonctionnalités du logiciel Max+plus II, nous allons maintenant nous intéresser à quelques fonctions avancées qui permettront d'optimiser la synthèse d'un FPGA.
Ce travail ne donnera qu'un aperçu rapide des importantes possibilités du logiciel.
Nous travaillerons dans l'optique de circuits cible de la famille MAX7000 et FLEX10K (c'est ce type de circuit dont nous disposons sur carte de développement). Pour une meilleure compréhension, il est fortement conseillé d'avoir fait une étude, même succincte des structures de ces deux familles de circuits.
Les fonctionnalités avancées ne sont intéressantes que pour des descriptions complexes ; pour une première approche nous nous limiterons cependant à des exemples très simples.

Nous verrons dans une première partie comment modifier le placement (et donc le routage) au sein d'un circuit et les conséquences sur les performances.
Dans une seconde partie, nous nous intéresserons à l'analyse des performances temporelle d'un circuit.
On rappelle qu'une aide est disponible à tout moment soit par la touche F1 (aide générale), soit en allant chercher en haut de l'écran l'icône d'aide contextuel (repéré par « ? ») et en cliquant sur l'outil, la fenêtre, le message etc… nécessitant un éclaircissement.

1 Influence du placement

Objectifs : apprendre à modifier le placement au sein d'un circuit, observer l'influence sur les performances de celui-ci.

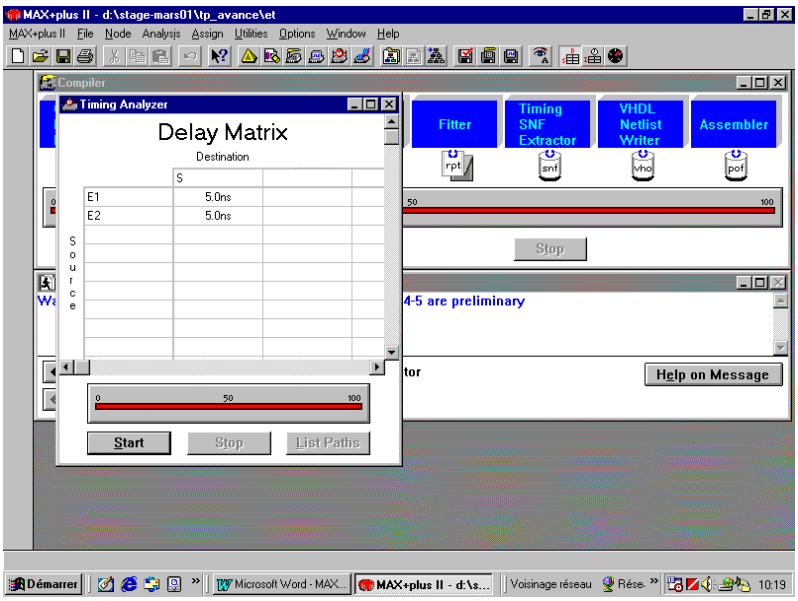
1.1 Insertion au sein d'un MAX7000

Considérons la description VHDL suivante :

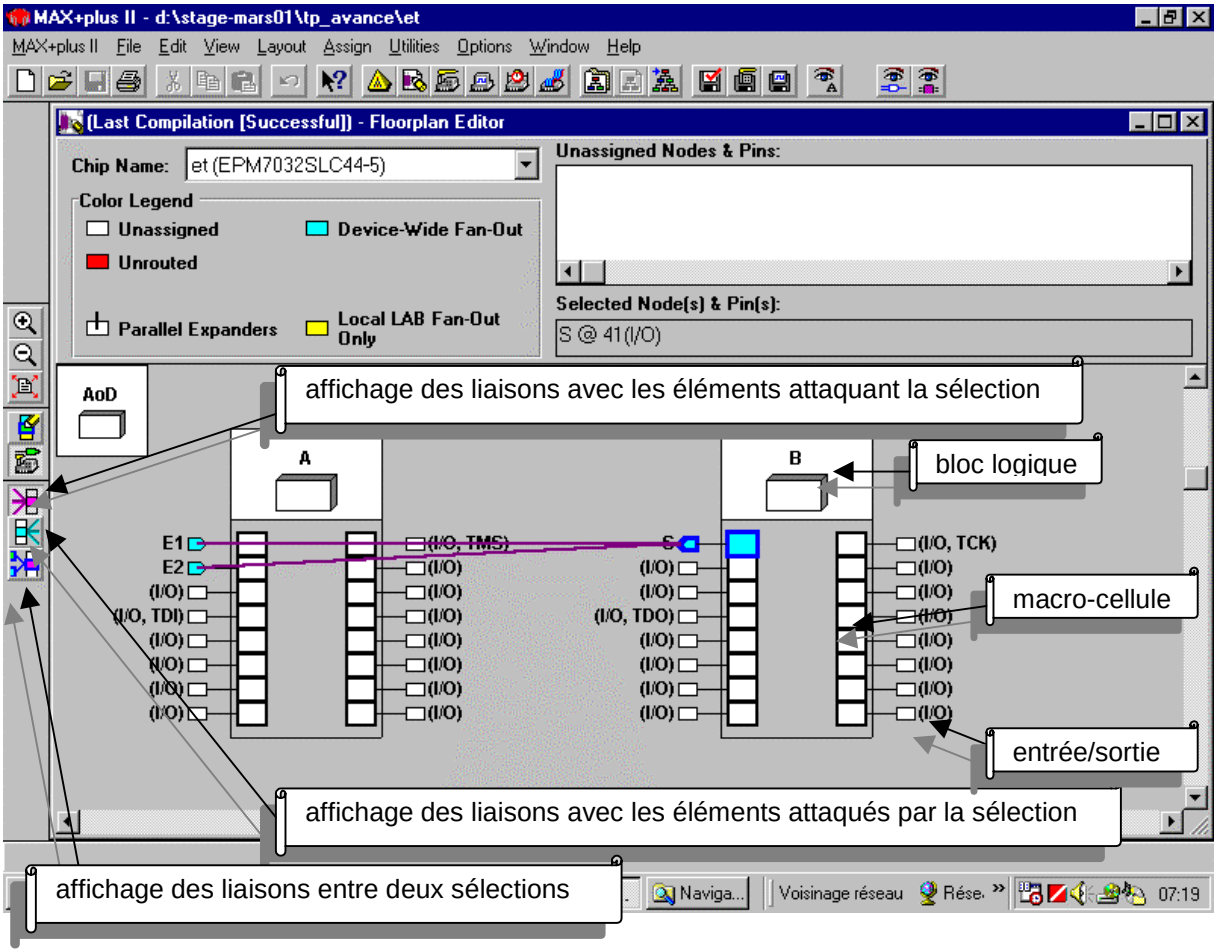
```
--------------------------------------------------------------------------------------------------------------------------------
entity ET is
        port    (       E1, E2          : in bit;
                        S               : out bit );
end ET;

architecture ARCH_ET of ET is
begin
        S <= E1 and E2;
end ARCH_ET;
--------------------------------------------------------------------------------------------------------------------------------
```

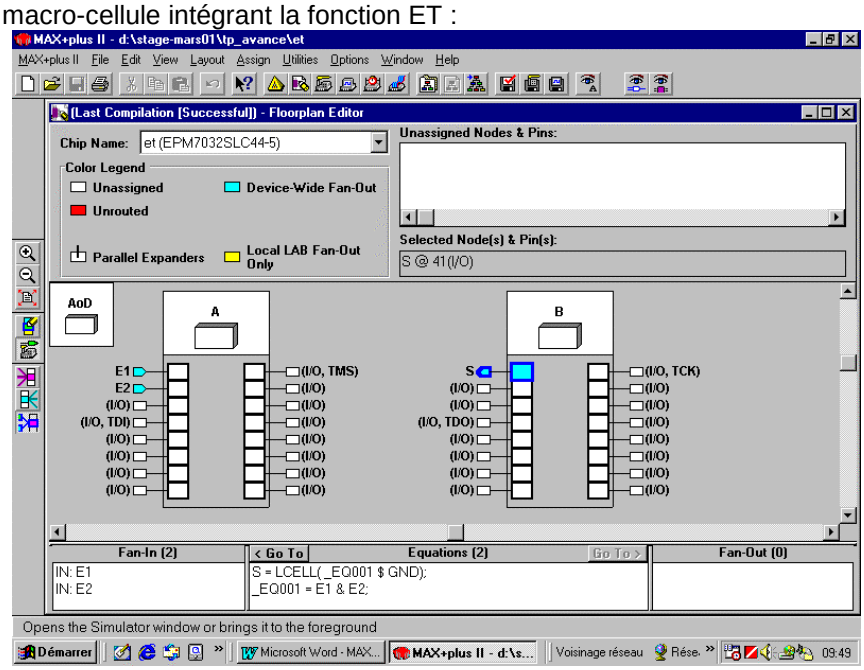
Comme nous pouvons le constater la fonction synthétisée est très simple, ce qui nous permettra de nous concentrer uniquement sur les fonctionnalités du circuit.
Compiler le projet en imposant un circuit de la série MAX7000S.
Observer dans le rapport de compilation les ressources utilisées et les équations générées. On pourra s'aider de la documentation du circuit afin de mieux comprendre le rapport da compilation.
Pourquoi le logiciel utilise t-il 2 macro-cellules, alors qu'une seule semblerait suffisante pour synthétiser cette fonction ET ?
Faire une analyse temporelle (**Max+plus II / Timing Analyzer**) pour noter les temps de propagation entre les entrées et sorties. Noter ces valeurs afin de pouvoir comparer par la suite.



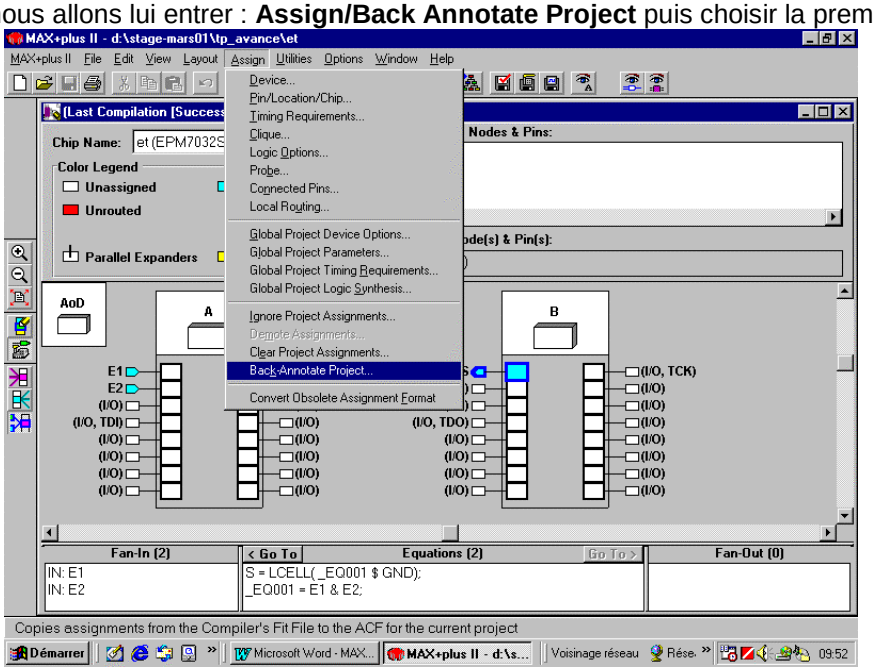
Ouvrir l'éditeur de placement (**Max+plus II / Floorplan Editor**) et observer le placement des macro-cellules et entrées/sorties utilisées. Si la vue de l'intérieur du circuit ne s'affiche pas, valider **LAB View** dans le menu **Layout** ou double clic gauche au milieu de l'écran.
Il est alors possible de visualiser les liaisons entre les différents éléments au moyen des icônes à gauche de l'écran (toutes ces possibilités sont également disponibles dans le menu **Option**) :



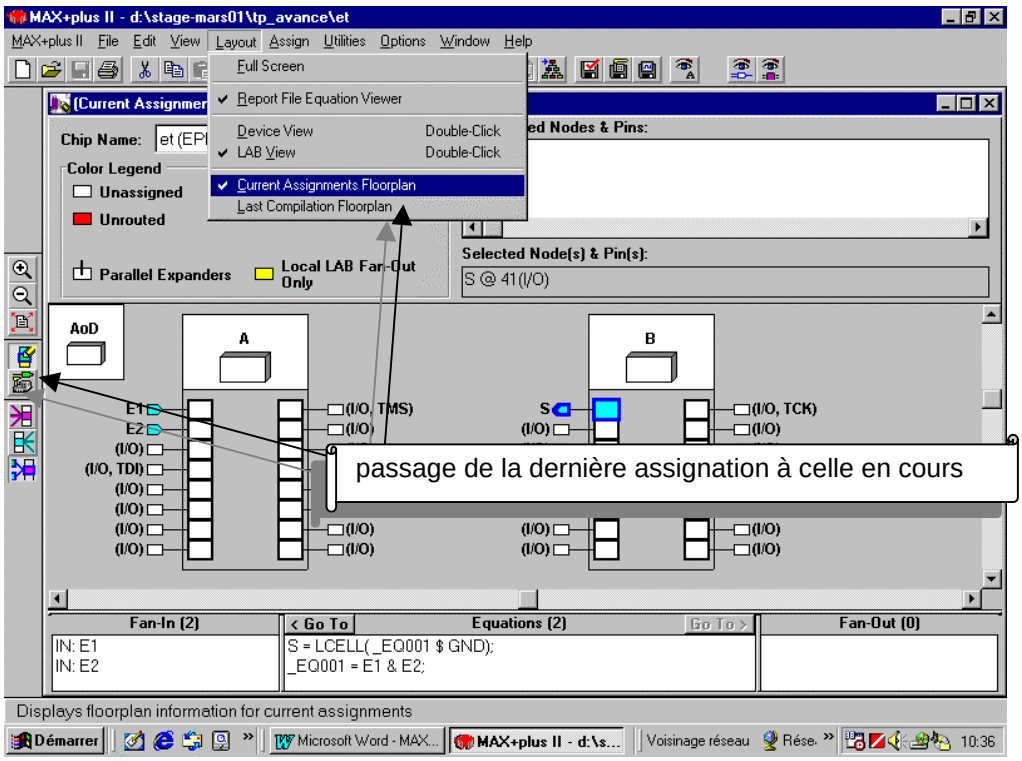
Pour sélectionner deux éléments à la fois, on sélectionne le premier avec la souris, puis le second de la même manière en gardant la touche Shift enfoncée.
On notera également les couleurs correspondant aux différentes ressources.
Il est possible de faire apparaître l'équation liée à chaque macro-cellule
Pour cela activer **Report File Equation Viewer** dans le menu **Layout**. L'écran suivant apparaît si on sélectionne la macro-cellule intégrant la fonction ET :



Nous allons maintenant modifier l'emplacement des macro-cellules choisies par le logiciel.
Essayer de déplacer une entrée par « glisser coller ». Comme on peut le voir, cette opération ne fonctionne pas. Il faut préciser avant au logiciel que l'on souhaite faire une « retro-annotation ».
En effet, la phase de placement routage est effectuée par le « fitter » lorsque la compilation est lancée. Il faut donc préciser au logiciel que l'on souhaite effectuer de nouveau cette étape avec les données que nous allons lui entrer : **Assign/Back Annotate Project** puis choisir la première option.



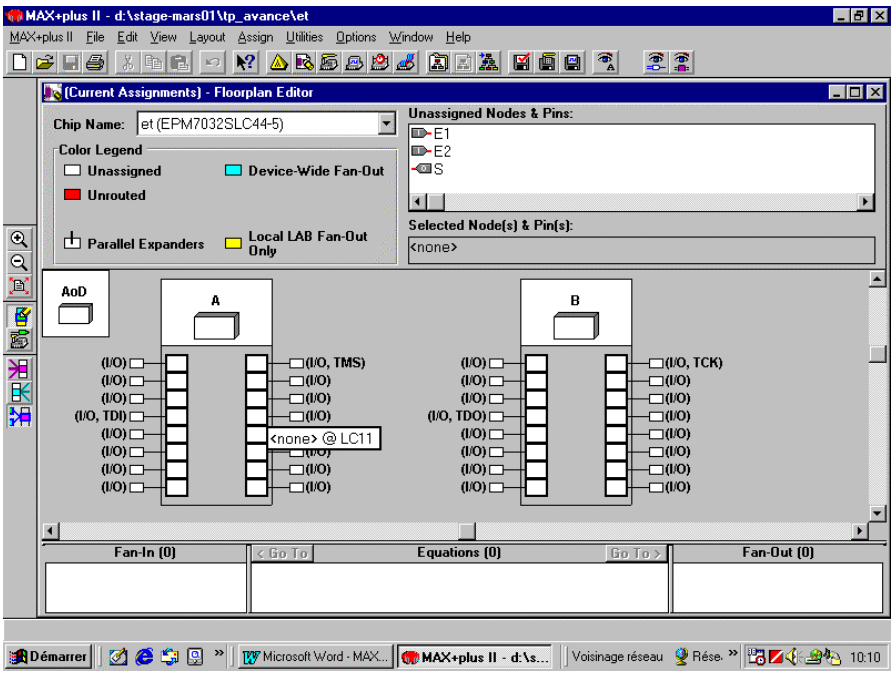
Il va maintenant y avoir deux vues du placement : celle qui vient d'être compilée (**Last Compilation Floorplan**) et celle que nous sommes éventuellement en train de modifier (**Current Assignment Floorplan**). On passe de l'une à l'autre par le menu **Layout** ou par une icône à gauche de l'écran.



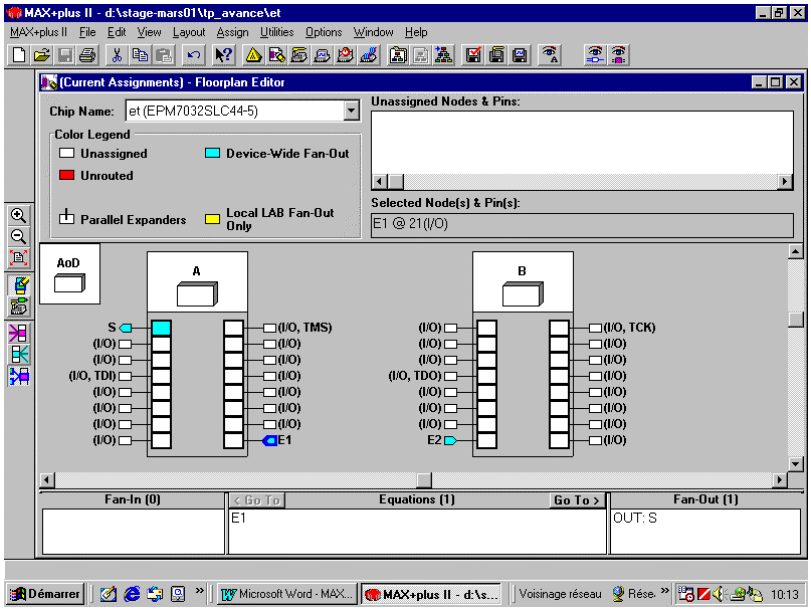
Lorsque l'on a activé l'assignation courante, on peut alors déplacer les entrées/sorties et les blocs par glisser coller. Les éléments déplacés peuvent être repérés en étant affichés en gris en sélectionnant **Show Moved Nodes In Gray** du menu **Option**.

On peut remarquer que si on déplace la macro-cellule sélectionnée sans la sortie, cette dernière disparaît.

Pour initialiser le placement : **Assign/Clear Project Assignment** , les entrées sorties apparaissent alors toutes dans la partie des nœuds non assignés.

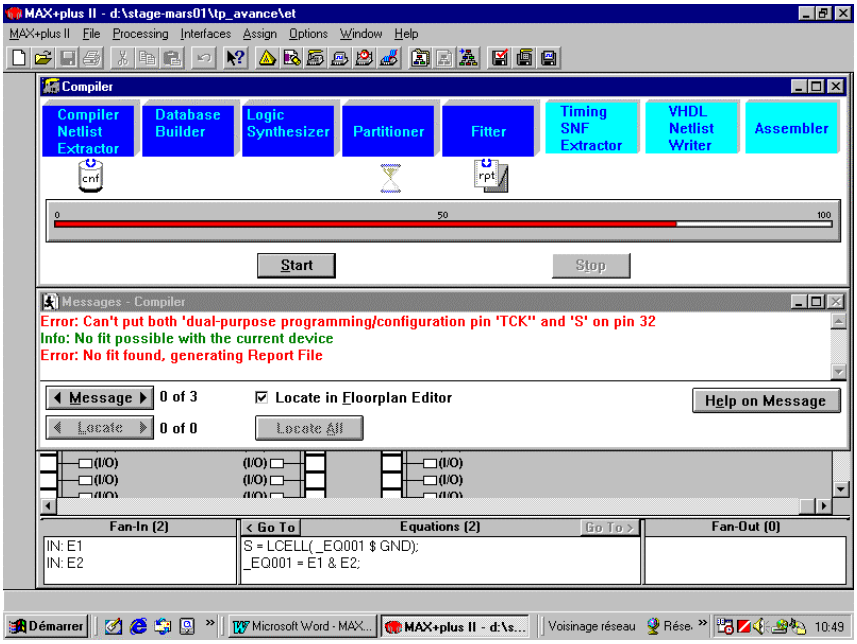


Affecter les différents éléments par glisser coller pour obtenir la configuration suivante (on placera la sortie sur une broche et non sur une macro-cellule) :



Relancer la compilation et vérifier dans le rapport de compilation que les modifications ont bien été prises en compte.
Lancer une analyse temporelle pour mesure les temps de propagation et comparer par rapport la précédente. Conclusion.

Nous allons maintenant affecter volontairement une borne illicite à une des entrées, ce qui nous permettra de tester les différents messages d'erreur générés par le logiciel.
A l'aide de l'éditeur de placement, affecter la sortie S sur la borne sur la broche TCK (en haut à gauche du bloc B), normalement réservée à la programmation.
Lancer la compilation, acquiescer aux différents messages d'erreur, puis sélectionner le premier message du compilateur en validant l'option **Locate in Floorplan Editor**.



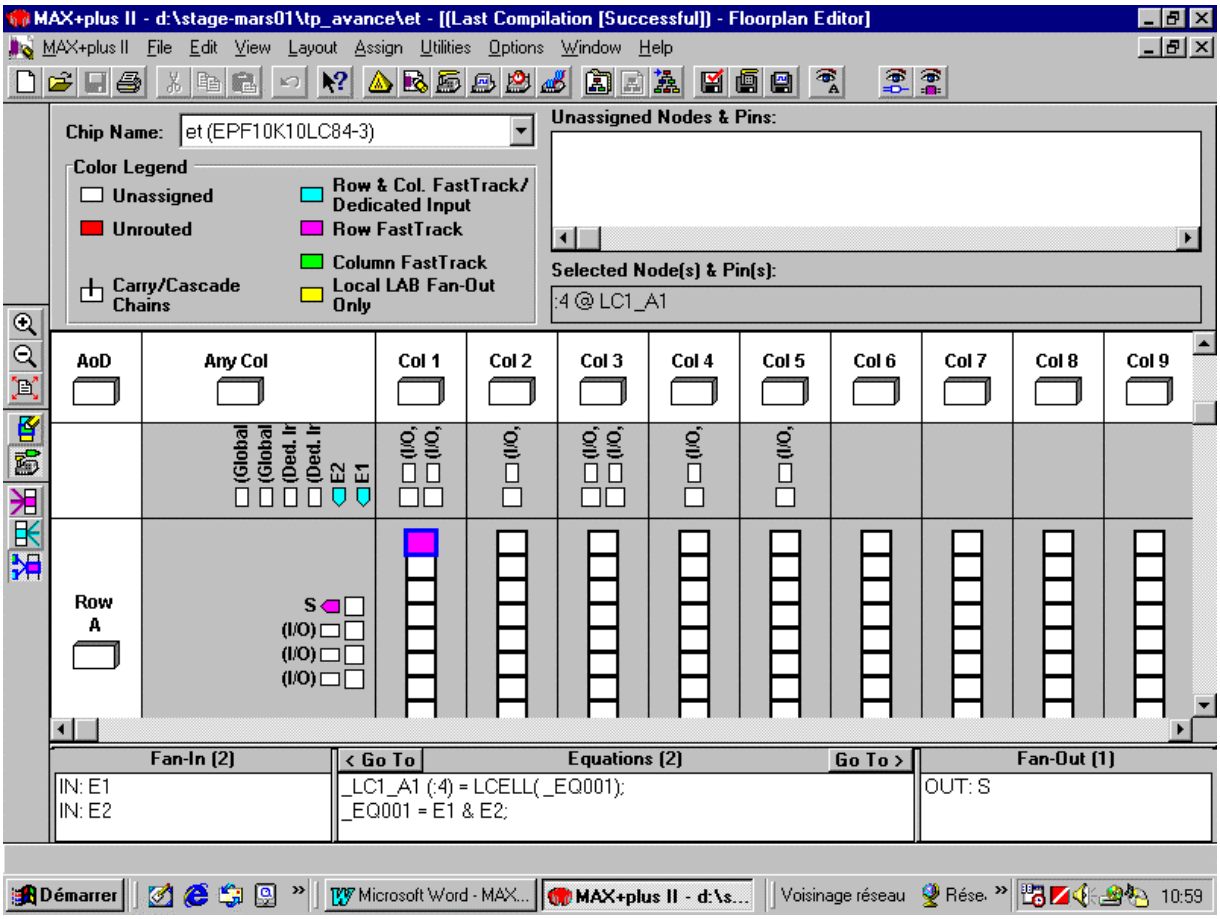
Lorsque la localisation est demandée, la broche incriminée est sélectionnée. Corriger l'erreur puis recompiler.

1.2 Insertion au sein d'un FLEX10K

Compiler la même description en imposant un circuit de la série Flex10K.

Lancer l'analyseur temporel. Que remarque t-on au niveau des temps de propagation ?

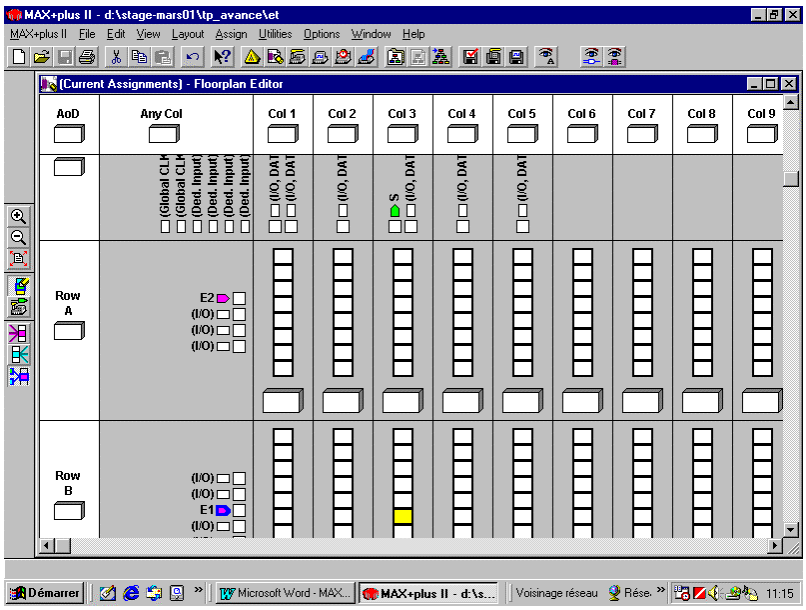
Ouvrir l'éditeur de placement, puis le rapport de compilation.



Etudier et comparer les informations fournies par les deux sources. On notera en particuliers la manière dont sont repérées les cellules logiques au sein du circuit : LC<n°cellule_dans_bloc>_<n°ligne><n°colonne>.

Nous allons maintenant changer les affectations. Pour cela déclarer une rétro-annotation. Sur une description aussi simple, il est bien évident que nous n'obtiendrons pas des performances supérieures à celles de l'affectation d'origine par le logiciel, et ce n'est pas notre but. Aussi, répondra t-on par la négative aux recommandations du logiciel.

Toutes les configurations n'étant pas possibles, on déplacera la cellule logique utilisée en LC7B3, la sortie en 8 et les entrées en 16 et 23 (l'écran suivant est obtenu en validant **Full Screen** dans **Layout**).



Lancer la compilation et afficher les temps de propagation. Conclusion.

2 L'analyseur temporel

Objectif : apprendre à caractériser un projet d'un point de vue des performances temporelles.

2.1 Performances temporelles

Pour illustration prenons comme exemple le compteur dont la description VHDL suit :

```
-------------------------------------------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use  ieee.std_logic_unsigned.all;

entity COMPTEUR0 is
        generic         (N: integer := 24);
        port    (H, RAZ_S, RAZ_AS    : in STD_LOGIC;
                S                    : out STD_LOGIC_VECTOR (N-1 downto 0) );
end COMPTEUR0;

architecture ARCH_COMPTEUR0 of COMPTEUR0 is
        signal X        : STD_LOGIC_VECTOR (N-1 DOWNTO 0);
        begin
                process (H, RAZ_AS)
                begin
                        if RAZ_AS = '1' then X <= conv_std_logic_vector (0,N);
                        elsif (H 'event and H = '1') then
                                if (RAZ_S='1') then X <= conv_std_logic_vector (0,N);
                                else X <= X + 1 ;
                                end if;
                        end if;
                end process;
                S <= X;
end ARCH_COMPTEUR0 ;
-------------------------------------------------------------------------------------------------------------------------
```

Il s'agit d'un compteur générique, dont la largeur de bus est initialisé à 4 bits, comprenant une remise à zéro synchrone et une asynchrone.
Compiler le projet en laissant le logiciel choisir un circuit cible dans la série MAX7000S.

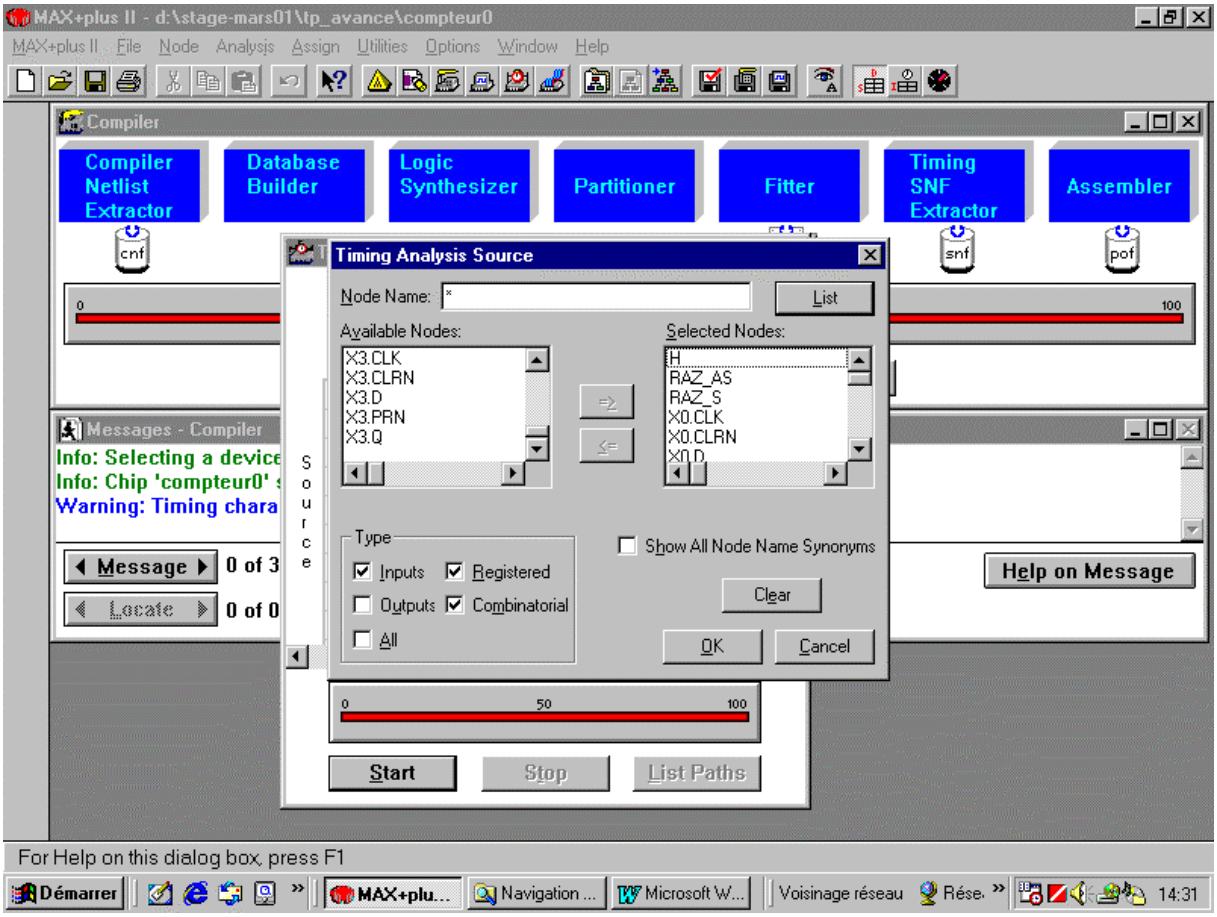
Etudier le rapport de compilation : le logiciel a utilisé des bascules D à entrée de validation (DFFE). ON pourra consulter l'aide pour savoir à quoi correspondent les équations et vérifier qu'on a bien un compteur synchrone classique à bacules D.

L'analyse des performances temporelle peut être évaluée à l'aide de plusieurs outils :
- le simulateur temporel dont nous avons vu l'utilisation lors de la séance précédente,
- l'analyseur de temps de propagation,
- l'analyseur de temps de prépositionnement et temps de maintient,
- l'analyseur de fréquence maximale.

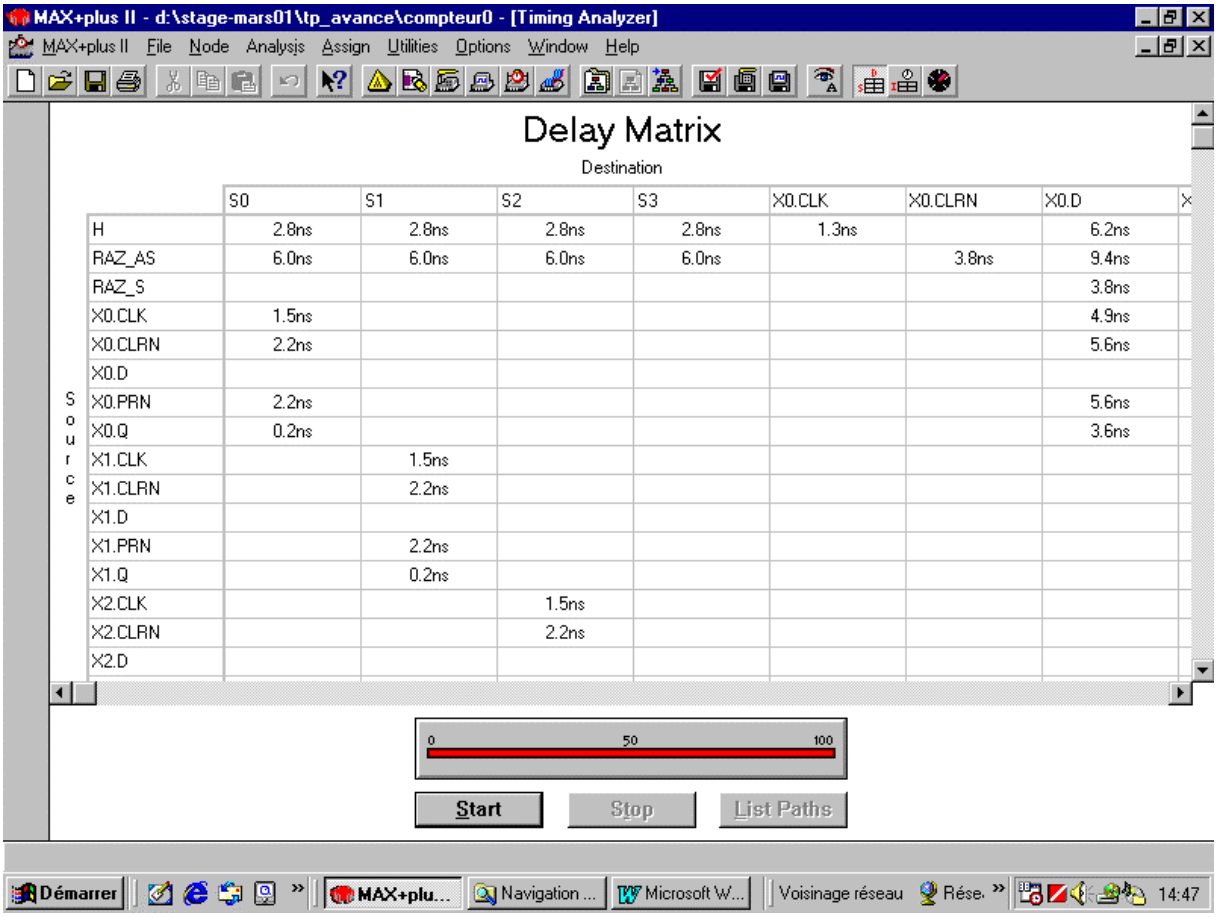
On accède aux trois derniers par l'analyseur temporel : **Max+plus II/Timing Analyseur**, le projet devant bien entendu avoir été compilé avec succès auparavant.

2.1.1 Analyseur de temps de propagation

Dans l'analyseur de temps, on y accède par **Delay Matrix** du menu **Analysis**.
Il va nous permettre de déterminer le délai entre l'application d'une commande en entrée et le résultat sur la sortie.
Les différentes entrées et sorties à prendre en compte sont définissables par le menu **Node** puis **Timing Analysis Source**.



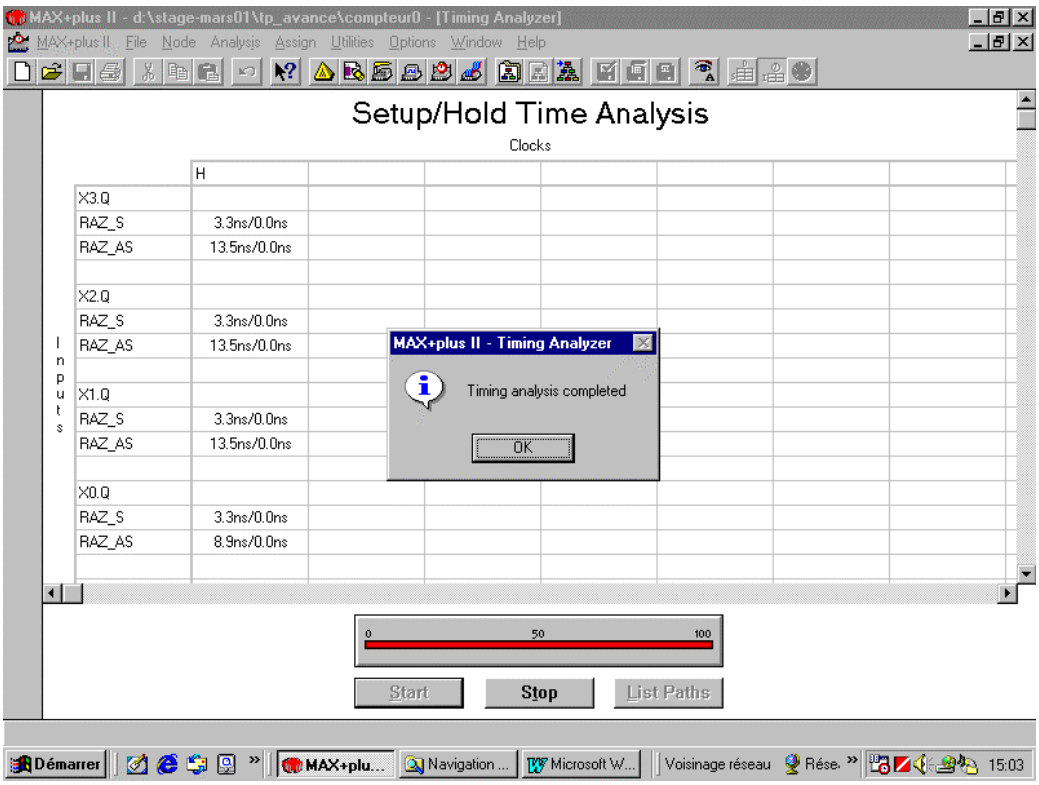
On pourra alors sélectionner comme paramètres source toutes les entrées, ainsi que les entrées et sorties des bascules internes (signaux X0.D, X0.CLK, X0.Q, X0.CLRN, X0.PRN, X1.D etc…).
On fera de même pour les signaux destination par le menu **Node** puis **Timing Analysis Destination** où on pourra cette fois sélectionner les signaux de sorties S et les signaux des bascules.
Supprimer éventuellement la désactivation de la prise en compte des signaux d'initialisation des bascules ; dans le menu **Option** désactiver **Cut Off Clear & Preset Path**.
Puis lancer la simulation.



Lorsqu'il existe un chemin direct (avec une bascule maximum) entre la source et la destination, un temps de propagations est affiché. Les temps maximum et minimum peuvent être affichés si plusieurs chemin sont possibles.
Le tableau tel que nous le voyons étant difficile à exploiter (même avec un projet simple), nous comprenons l'intérêt de pouvoir sélectionner les sources et destinations.

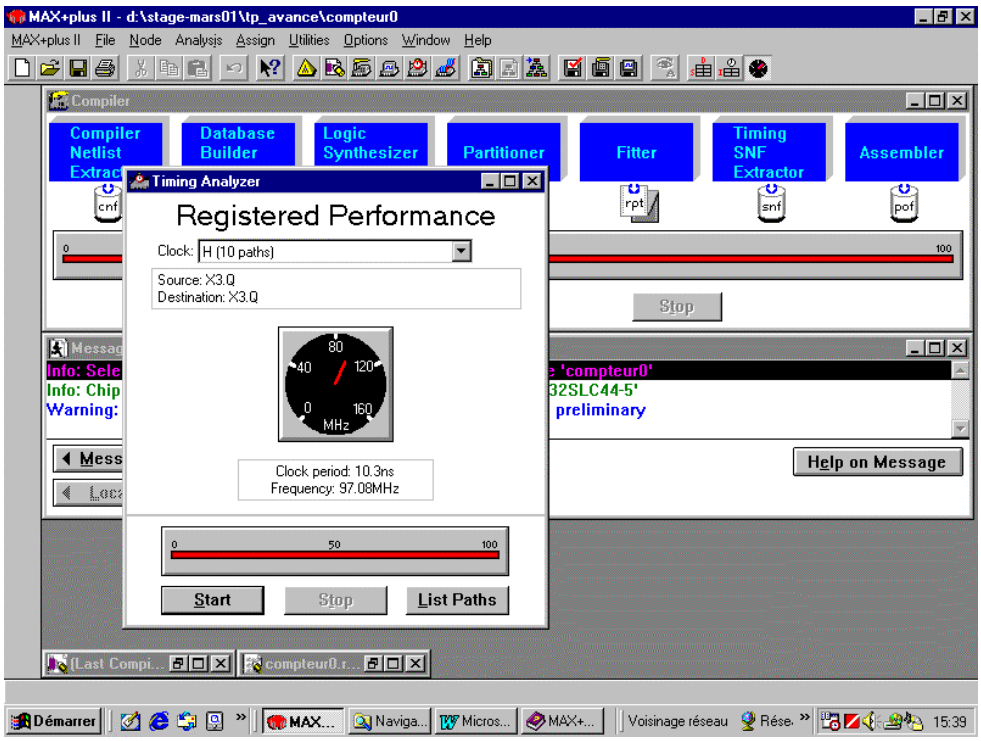
2.1.2 Analyseur de temps de prépositionnement et de maintien

Sur une bascule à front, les entrées doivent être stables un temps dit de prépositionnement (set-up time) avant le front actif de l'horloge et un temps dit de maintien (hold time) après.
On conçoit qu'il en aille de même pour toutes les entrées synchrones.
Cette notion est étendue aux verrous à niveau et aux entrées asynchrones.
L'outil que nous étudions maintenant permet de déterminer ces temps. On y accède dans l'analyseur temporel par **Setup/Hold Matrix** du menu **Analysis**.
On initialise les sources et destinations de la même manière que précédemment. Les destinations correspondent à toutes les bascules. Les sources pourront être toutes les entrées de la description, elles seront listées à gauche du tableau si elles attaquent une entrée de donnée ou d'initialisation, et en haut du tableau si elles attaquent une horloge.



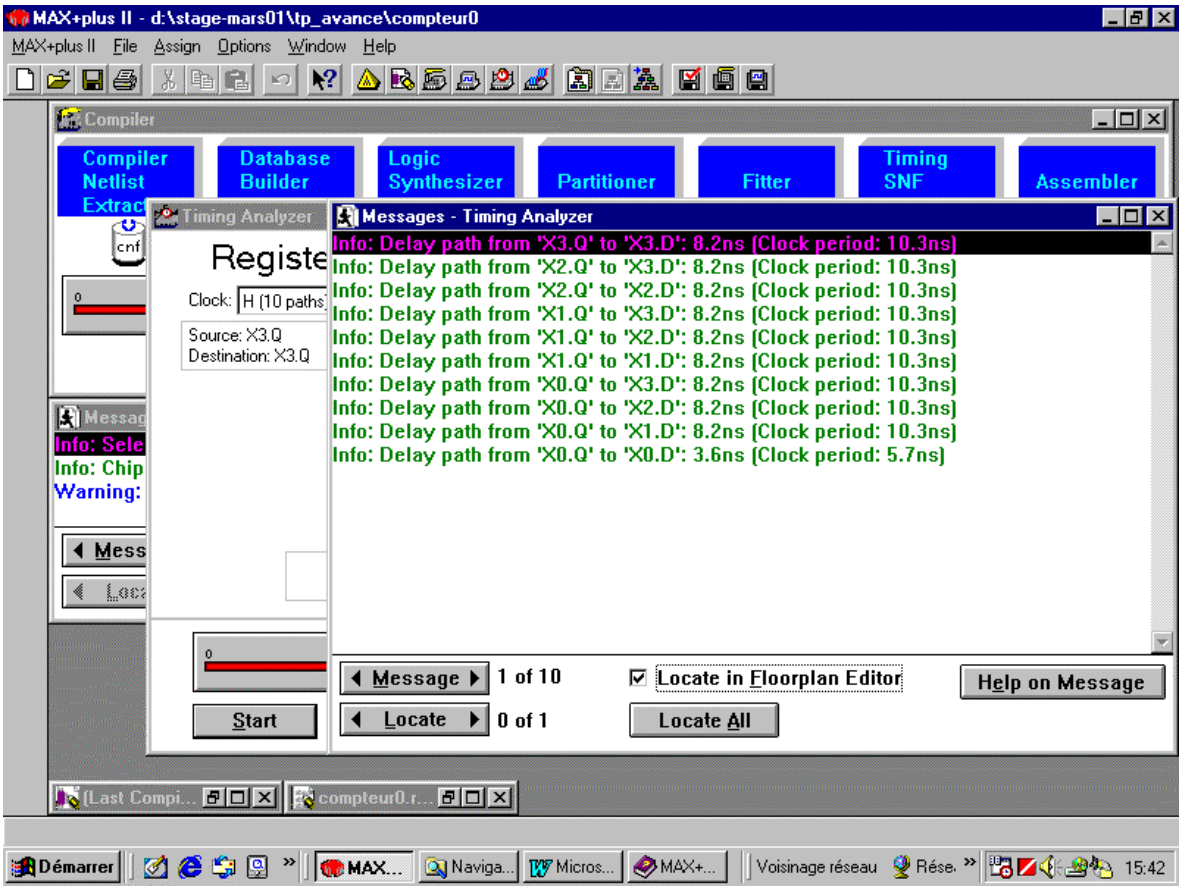
2.1.3 Analyseur de fréquence maximale

Dans l'analyseur de temps, on y accède par **Registered Performance Matrix** du menu **Analysis**. Il permet de donner la fréquence maximale de fonctionnement.



2.1.4 Localisation d'un chemin critique

Quel que soit l'outil utilisé, il est possible de visualiser dans l'éditeur de placement un chemin critique en cliquant sur **List Paths** et en sélectionnant le chemin affiché.



Valider ensuite l'option **Locate in Floorplan Editor** puis activer **Locate**.

Les macro-cellules concernées apparaissent alors dans l'éditeur de placement. Dans le cas de notre circuit cette option présente peu d'intérêt comme nous l'avons vu au paragraphe précédent. Dans le cas d'un FPGA, peut être sera t-il possible d'optimiser ainsi les performances dynamiques.

2.2 Influences de la complexité de la fonction et du choix du circuit sur la vitesse

Il n'est pas facile de caractériser un circuit en terme de vitesse, car ces performances vont dépendre pour un même circuit cible de la complexité de la fonction à synthétiser et de la manière dont le logiciel va l'implanter.

Rappeler pour la description précédente les fréquences maximales obtenues pour un circuit de la famille MAX7000S et pour FLEX10K (refaire les compilations si nécessaire). Noter à chaque fois le circuit choisi par le logiciel.

Synthétisons maintenant un circuit plus complexe par l'intermédiaire de ce compteur 24 bits prépositionnable ci-après.

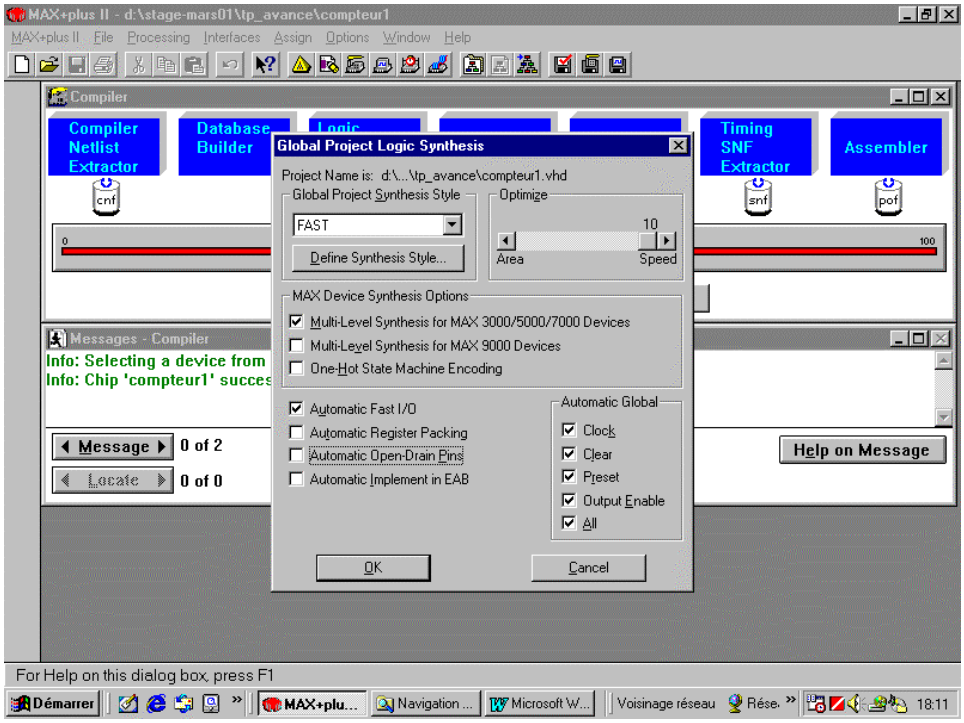
```
--------------------------------------------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```
use  ieee.std_logic_unsigned.all;

entity COMPTEUR1 is
        generic (       N: integer := 24;
                        M: integer := 16000000 );
                port(H, RAZ_S, RAZ_AS, OE, LD       : in      STD_LOGIC;
                        E                           : in STD_LOGIC_VECTOR (N-1 downto 0);
                        S                           : out STD_LOGIC_VECTOR (N-1 downto 0) );
end COMPTEUR1;

architecture ARCH_COMPTEUR1 of COMPTEUR1 is
        signal X        : STD_LOGIC_VECTOR (N-1 DOWNTO 0);
        begin
        process (H, RAZ_AS)
                begin
                if RAZ_AS = '1' then X <= conv_std_logic_vector (0,N);
                elsif (H 'event and H = '1') then
                        if (RAZ_S='1'or X>=M ) then X <= conv_std_logic_vector (0,N);
                        elsif LD='1' then X<=E;
                        else X <= X + 1 ;
                        end if;
                end if;
        end process;
        S <= X when OE='1' else (others =>'Z');
end ARCH_COMPTEUR1 ;
```

On effectuera la synthèse pour un circuit MAX7000S puis FLEX10K en optimisant soit la vitesse, soit la place occupée dans le circuit. Pour cela on choisira l'option **Normal** ou **Fast** de la boite de dialogue **Global Project Logic Synthesis** de l'option **Global Project Synthesis Style**. Dans le cas du MAX7000S on validera l'option de synthèse à plusieurs passes.



Conclusion ?