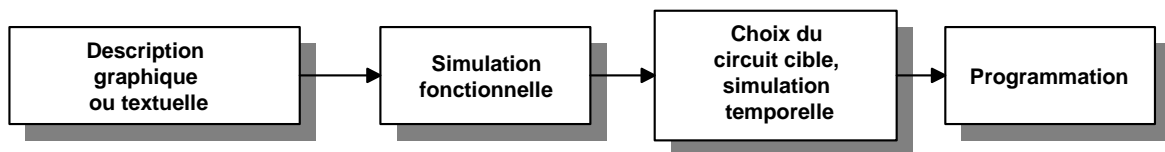


Programmer un FPGA avec Quartus II

Comme nous l'avons vu, la conception de cartes électroniques est aujourd'hui fortement assistée par ordinateur.

Le logiciel Quartus II est dédié à la programmation des CPLD et FPGA du fabricant Altera ; il permet la description d'un projet, sa compilation, sa simulation logique et temporelle, son analyse temporelle et la programmation d'un circuit cible.



Le composant que nous programmerons, un FPGA de type «EP2C35F672C6 » de la famille Cyclone II se trouve sur une carte de développement, associé à des composants annexes.

1 Configurer Quartus et vérifier la validité de la licence

Avant d'utiliser le logiciel, il faut le configurer et vérifier la validité de la licence. Cette opération est normalement déjà réalisée. On se reportera éventuellement aux annexes 1 et 2 ;

2 Fonctions de bases

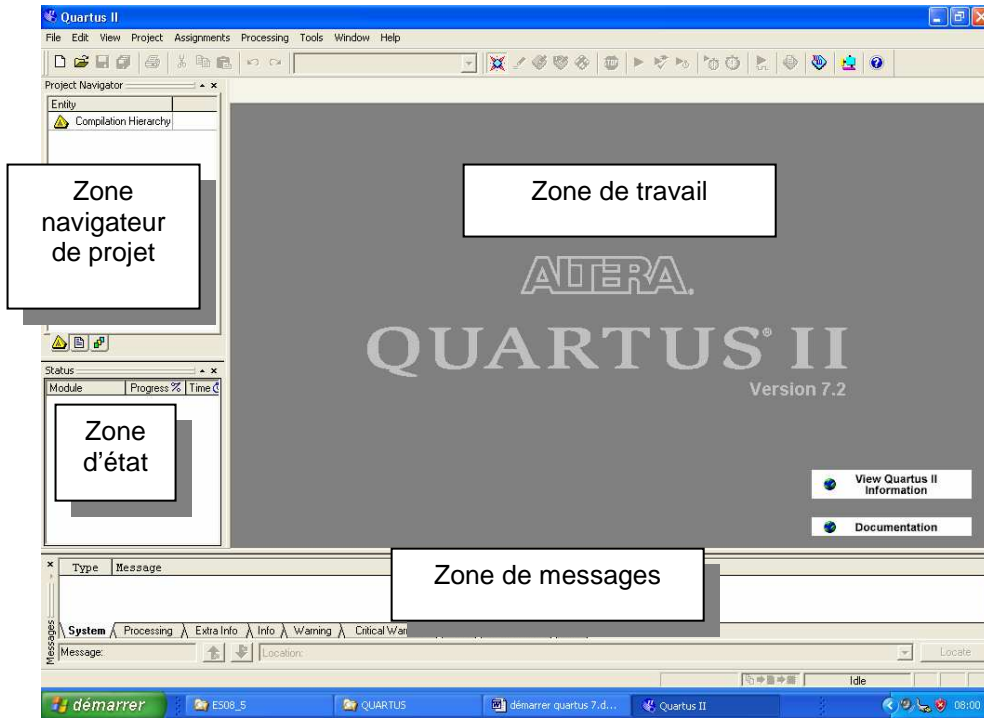
Dans cette partie, nous verrons comment décrire un projet, effectuer une simulation fonctionnelle, affecter les entrées sorties et programmer notre circuit cible.

Dans un premier temps, nous nous contenterons d'implanter des fonctions logiques de base au sein de notre FPGA. Quelle que soit la simplicité de notre conception, il est nécessaire d'ouvrir un nouveau projet

2.1 Ouvrir un nouveau projet

Ouvrir Quartus II comme n'importe quel logiciel, il présente alors l'interface suivant, comprenant 4 zones ou fenêtres principales :

Programmer un FPGA avec Quartus II

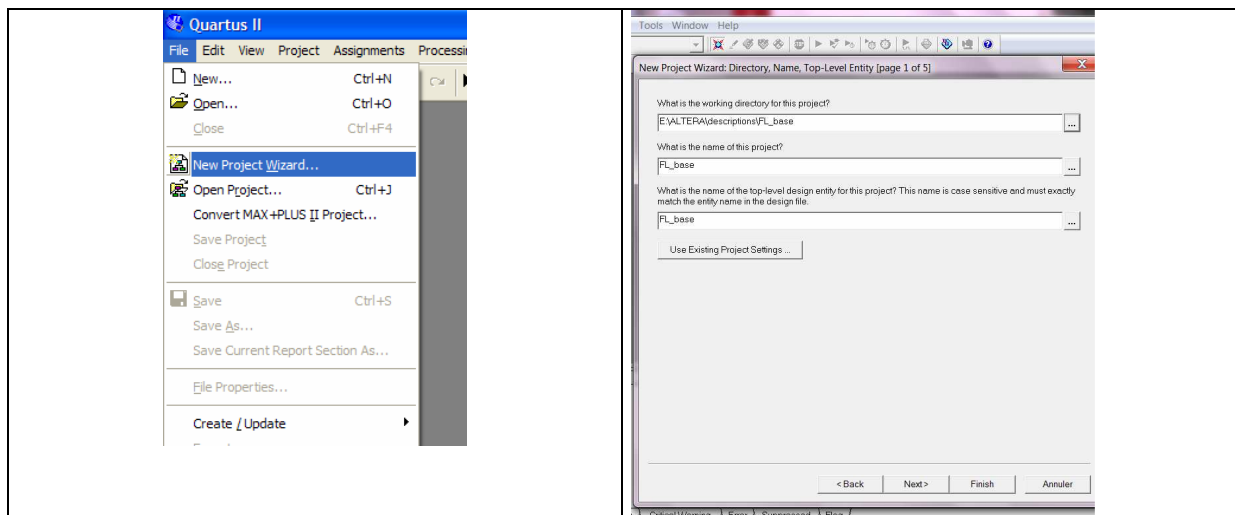


- une zone « navigateur de projet » permettant de gérer les différents fichiers d'un projet ;
- une zone de travail permettant la synthèse du projet ;
- une zone d'état (Status) permettant de voir l'avancement de la tâche en cours ;
- une zone de messages.

Si ces fenêtres n'apparaissent pas, on peut les activer par :



Après l'ouverture du logiciel, un assistant vous permet, après une fenêtre d'introduction, de commencer à configurer vos projets :



Programmer un FPGA avec Quartus II

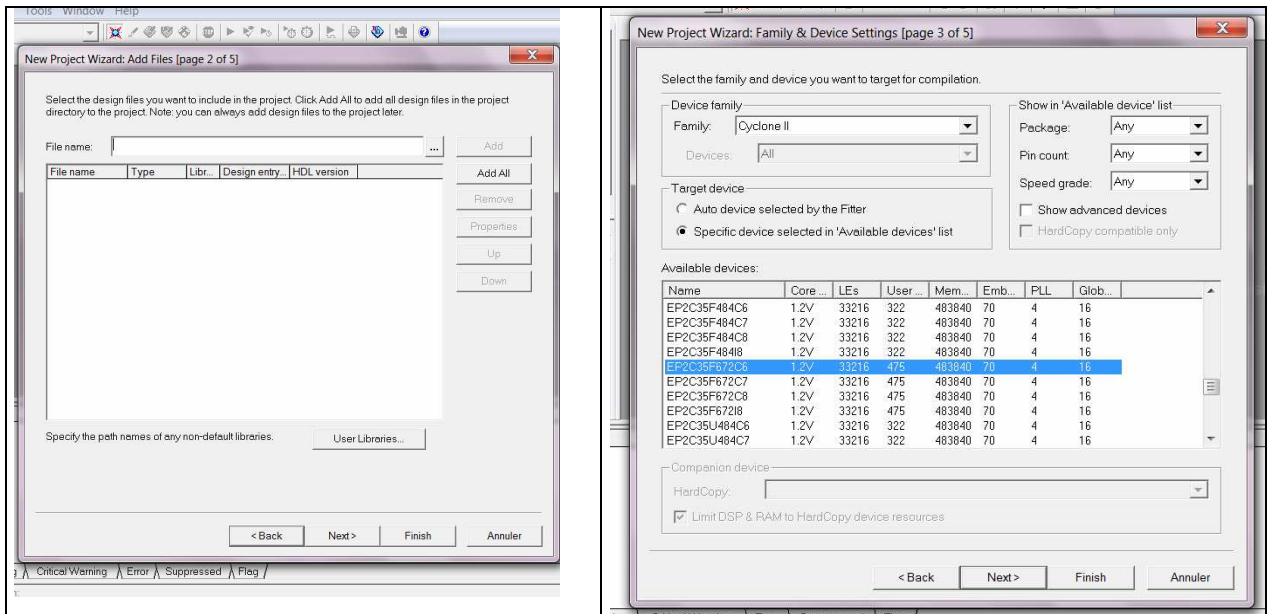
Choisir un répertoire de travail (mémoriser son emplacement) et définir un nouveau dossier à

l'aide de l'explorateur qui s'ouvre (FL_base par exemple) et attribuer un dossier nom à votre projet (qui peut être aussi FL_base pour faire simple).

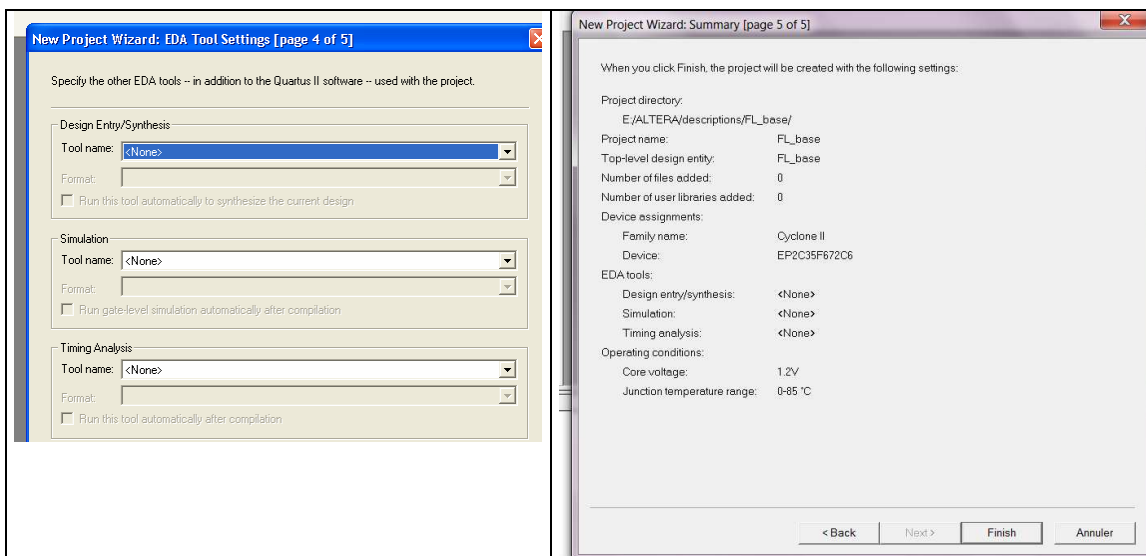
Attention : ne pas utiliser de caractères accentués pour les noms de projet et fichiers.

L'écran suivant permet éventuellement d'associer des fichiers au projet ; nous n'ajouterons rien de particulier.

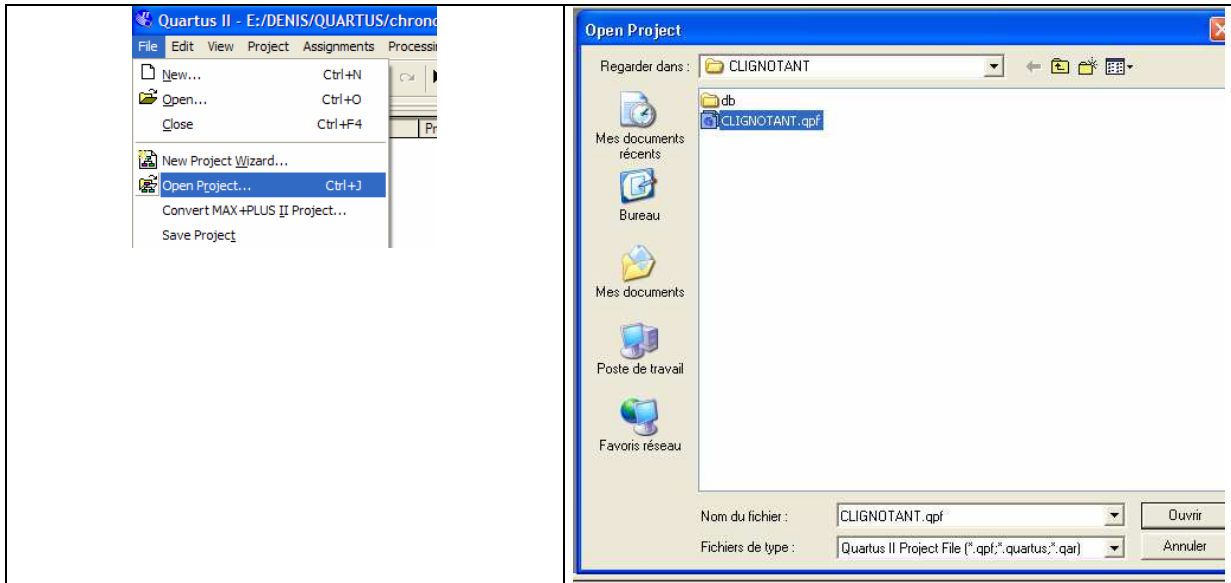
On choisit ensuite le circuit cible un «**EP2C35F672C6**» de la famille **Cyclone II**. Noter que cette étape permet également de définir le circuit cible par des critères tels que famille, brochage etc... ou de laisser libre choix au logiciel de choisir le circuit le mieux approprié au projet dans une famille donnée.



L'assistant permet également de sélectionner des outils EDA (Electronic Design Automation tools) autres que Quartus pour la synthèse, la simulation ou l'analyse temporelle. Nous nous contenterons des outils fournis par Quartus. Enfin, la dernière fenêtre résume les choix opérés.



Lorsque l'on souhaitera ouvrir de nouveau ce projet ou un autre, on procédera comme suit :

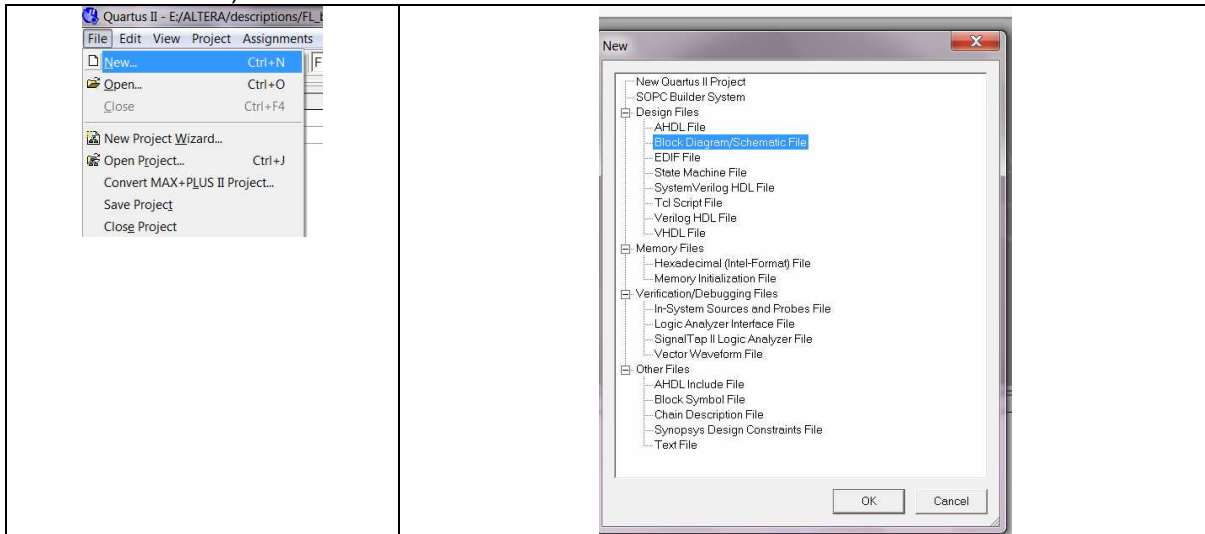


Dans Quartus, un fichier n'est rien s'il n'est pas intégré - et ouvert- dans un projet.

2.2 Description graphique d'un projet

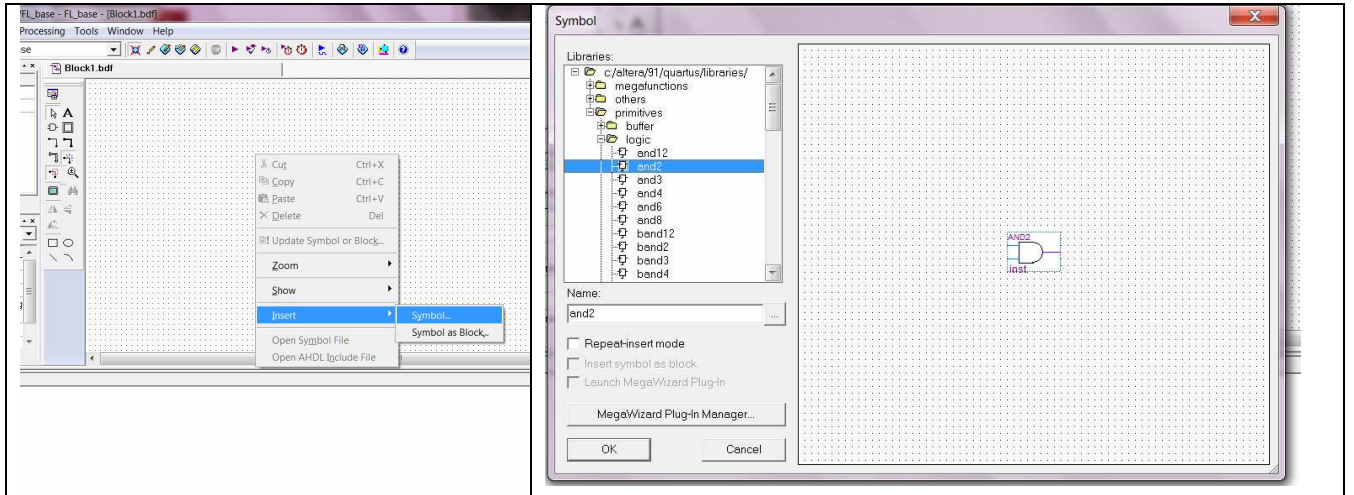
Décrivons une simple fonction ET pour notre projet

Pour cela, au sein de notre projet FL_base, ouvrons une feuille graphique (« **Block diagram / Schématic File** »).

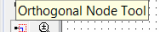


Dans la feuille qui vient de s'ouvrir, par un clic droit au milieu, insérer le symbole d'une fonction ET à deux entrées, « **and2** » que l'on trouvera dans la bibliothèque « **primitives / logic** »

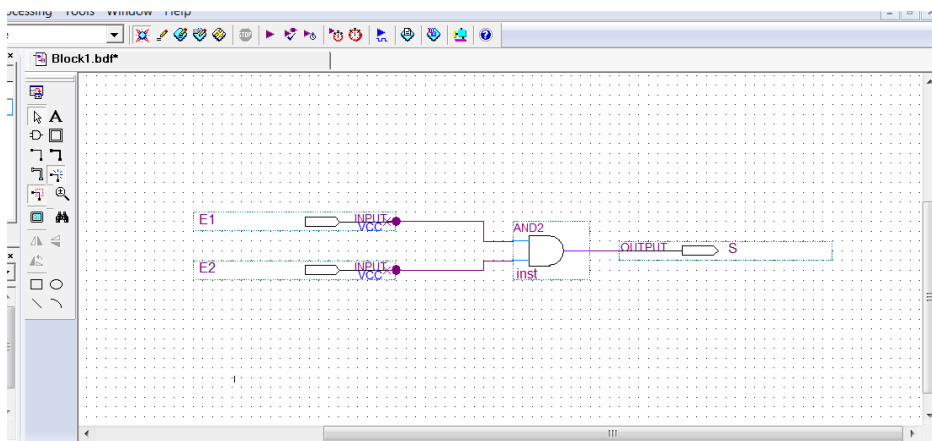
Programmer un FPGA avec Quartus II



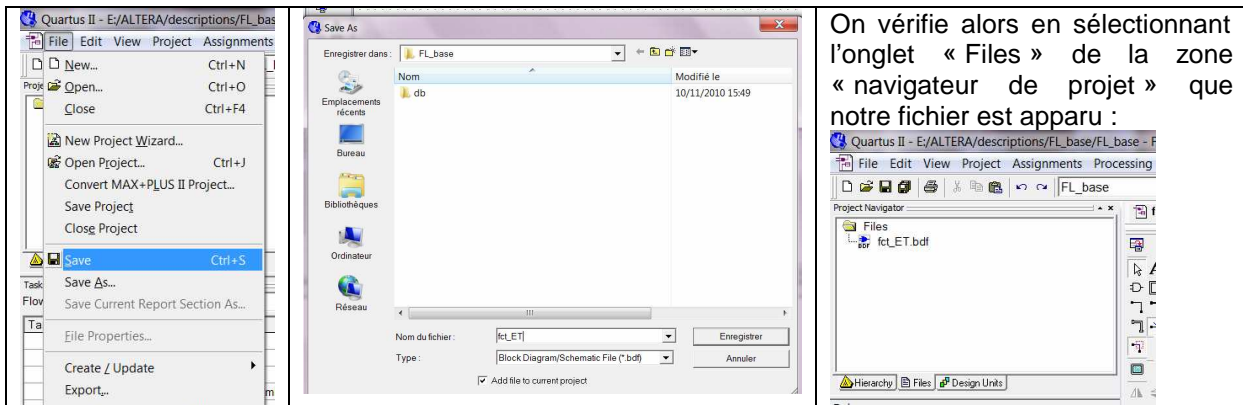
Placer ensuite deux entrées et une sortie avec les composants « **input** » et « **output** » qui se trouve dans le répertoire « **primitives / pin** », ainsi que les fils de liaisons à l'aide de l'icône « **Orthogonal**

Node Tool » à gauche de la feuille , puis donner un nom aux entrées et sortie en cliquant sur le nom par défaut, pour obtenir le schéma suivant :

A l'aide de l'outil loupe par clic droit ou clic gauche on peut agrandir ou compresser la zone d'affichage.



Sauvegarder ensuite le schéma, sous le nom par exemple de fct_ET ; votre schéma sera alors sauvegardé sous le nom fct_ET.bdf (pour Block Description File) au sein du projet FL_base qui peut éventuellement contenir plusieurs schémas ou descriptions diverses, correspondant soit à plusieurs versions d'un même système, soit à des descriptions interdépendantes les unes des autres.

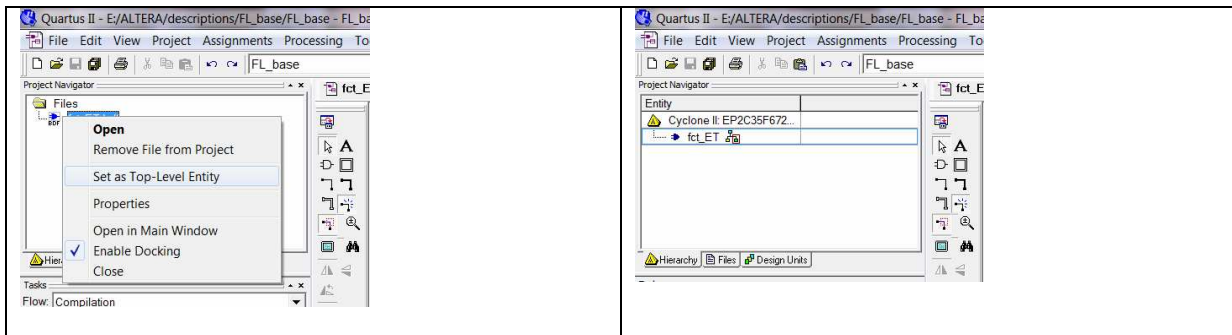


Un projet au sein de Quartus II pouvant gérer plusieurs descriptions, il est important de placer celle sur laquelle on va travailler au niveau le plus élevé de la hiérarchie du projet, ce qui peut se faire :

- Soit avec « **Set as Top Level Entity** » du menu contextuel du fichier concerné dans la zone de navigation ;
- Soit en cliquant dans la page correspondant au schéma puis « **Project -> Set as top Level Entity** » de la barre de menu ;
- Soit cliquant dans la page correspondant au schéma puis le raccourci clavier « **Ctrl Shift J** ».

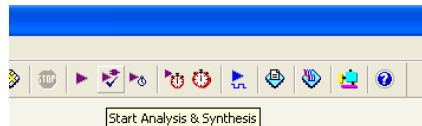
Avant d'entreprendre l'analyse d'une description ou sa compilation, toujours vérifier que celle-ci est bien au niveau de hiérarchie le plus élevée ; il est fortement conseillé de faire un « Ctrl Shift J » après avoir cliqué dans la page correspondant à la description pour l'activer.

On vérifie ensuite, dans la zone de navigation, avec l'onglet « **Hierarchy** » que notre fichier se trouve bien au niveau le plus élevé (pour l'instant, c'est le seul fichier, ce donc pas très compliqué...)



2.3 Analyse et synthèse

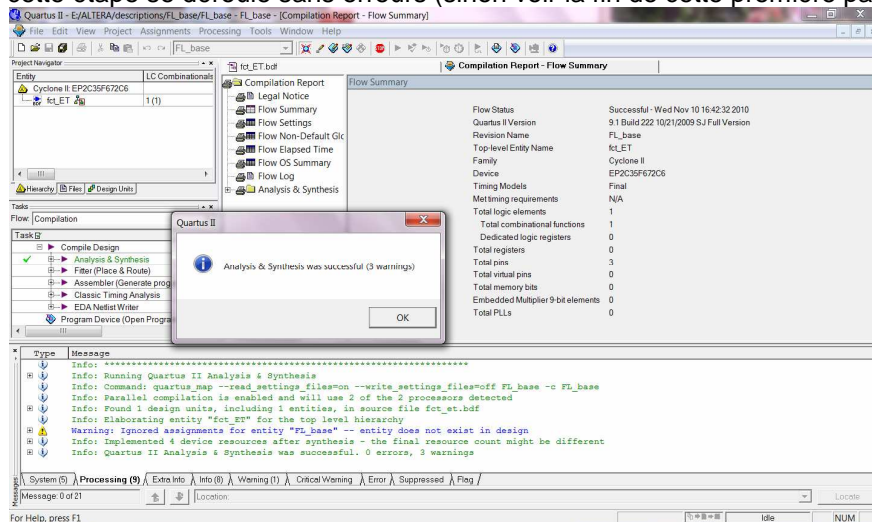
On peut ensuite lancer une première analyse du projet par l'icône appropriée :



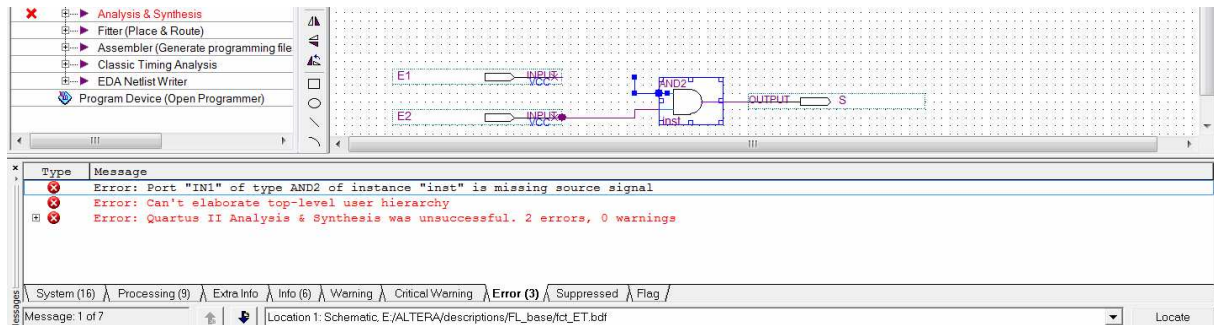
Cette première étape nous permettra entre autres :

- d'effectuer une simulation fonctionnelle de notre description ;
- de déclarer les entrées sorties au logiciel pour une affectation ultérieure.

Normalement cette étape se déroule sans erreurs (sinon voir la fin de cette première partie) :



En cas d'erreur, ouvrir l'onglet « Error » dans la fenêtre de message, faire un double clic sur la première erreur, ce qui aura pour effet d'afficher en bleu la zone du schéma posant problème.

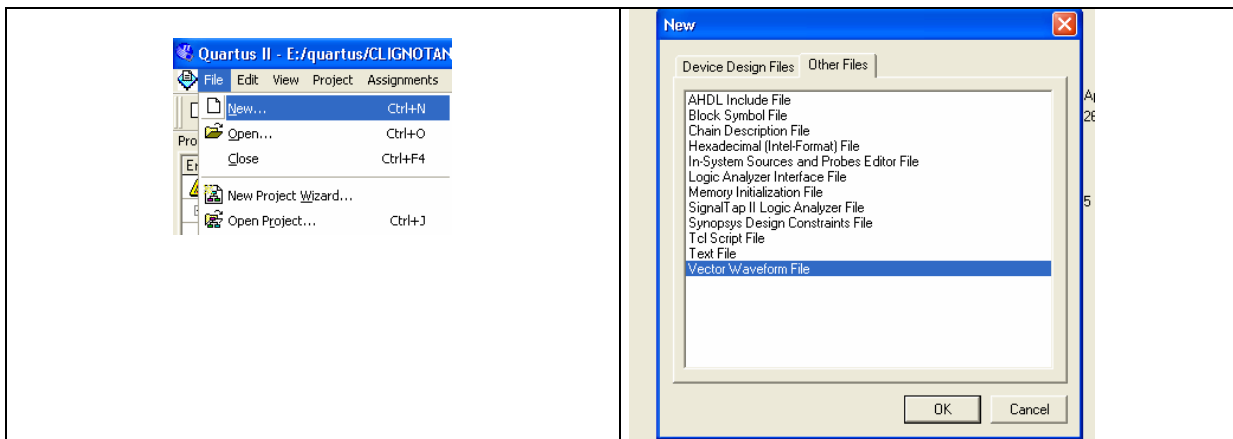


2.4 Simuler

Cette étape va nous permettre de valider le fonctionnement de notre description, d'un point de vue logique uniquement (sans prendre en compte les retards que va introduire le circuit cible). Avant de simuler, il est nécessaire de préciser les sorties (ici il n'y en a qu'une) que l'on souhaite observer, ainsi que les entrées que l'on applique.

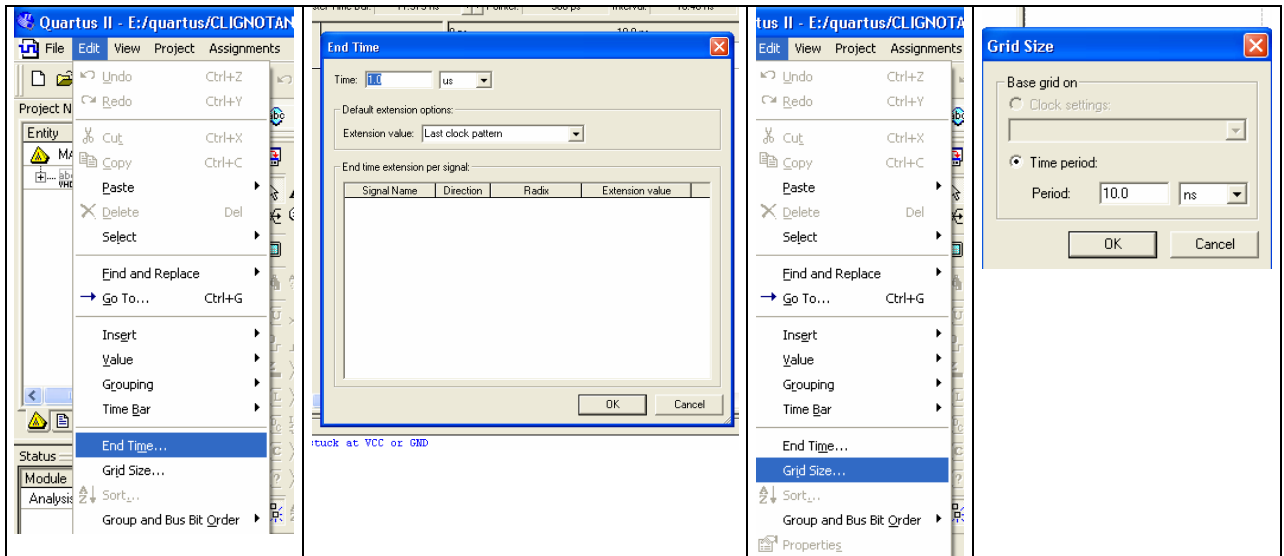
2.4.1 Créer et décrire le fichier de simulation

C'est un fichier graphique où sont décrits, sous forme de chronogrammes, les signaux d'entrée, ainsi que les signaux de sortie que l'on souhaite visualiser.

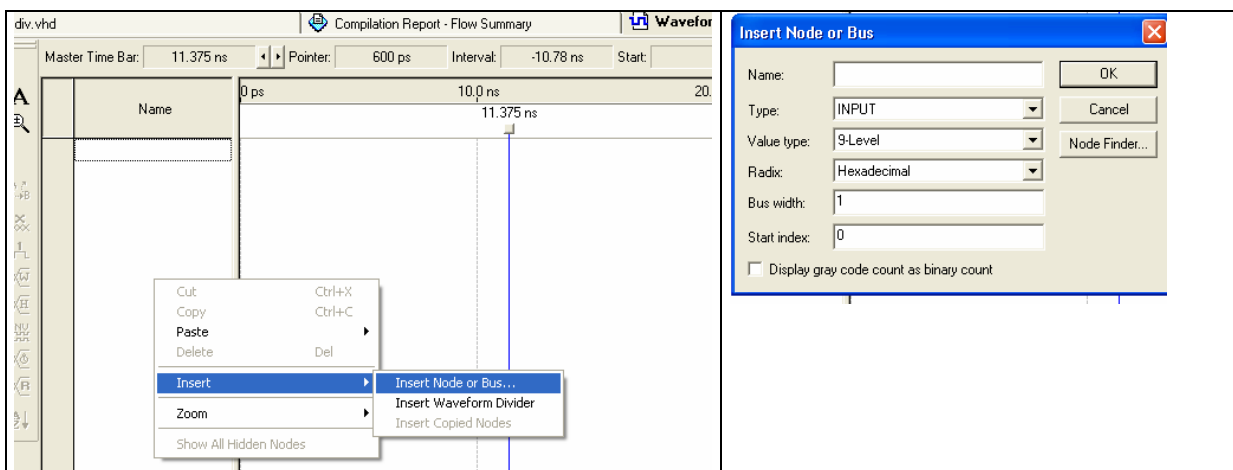


Préciser ensuite la durée de simulation (1 μ s par exemple), ainsi que le pas de la grille (10 ns par exemple).

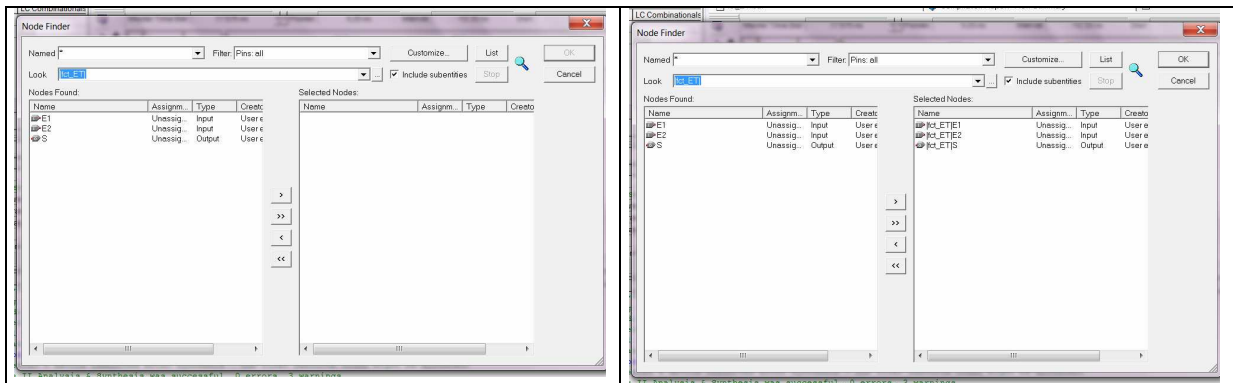
Programmer un FPGA avec Quartus II



On insère ensuite les différentes entrées sorties par un clic droit dans la colonne « **Name** » et « **Insert** » :



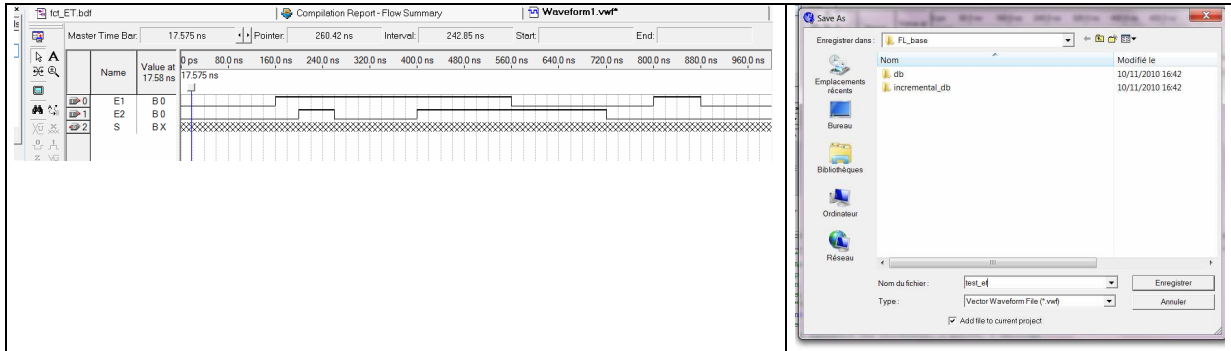
Dans la fenêtre qui s'ouvre, cliquer sur « **Node Finder...** », puis valider « **Pin all** » dans le filtre, et cliquer sur « **List** » : Sélectionner ensuite les signaux E1, E2 et S et les faire passer dans la fenêtre de droite.



On valide ensuite dans la dernière fenêtre.

A l'aide de l'outil loupe par clic droit ou clic gauche on peut agrandir ou compresser la zone d'affichage.

On impose ensuite l'allure des entrées (ici l'horloge E1 et E2) en cliquant sur le nom du signal, puis en utilisant les icônes à gauche : définir des temps à l'état 1 ou 0 en sélectionnant la zone du chronogramme concernée par un « glissé » de la souris bouton gauche enfoncé, puis en validant avec l'icône appropriée, pour obtenir ceci par exemple :

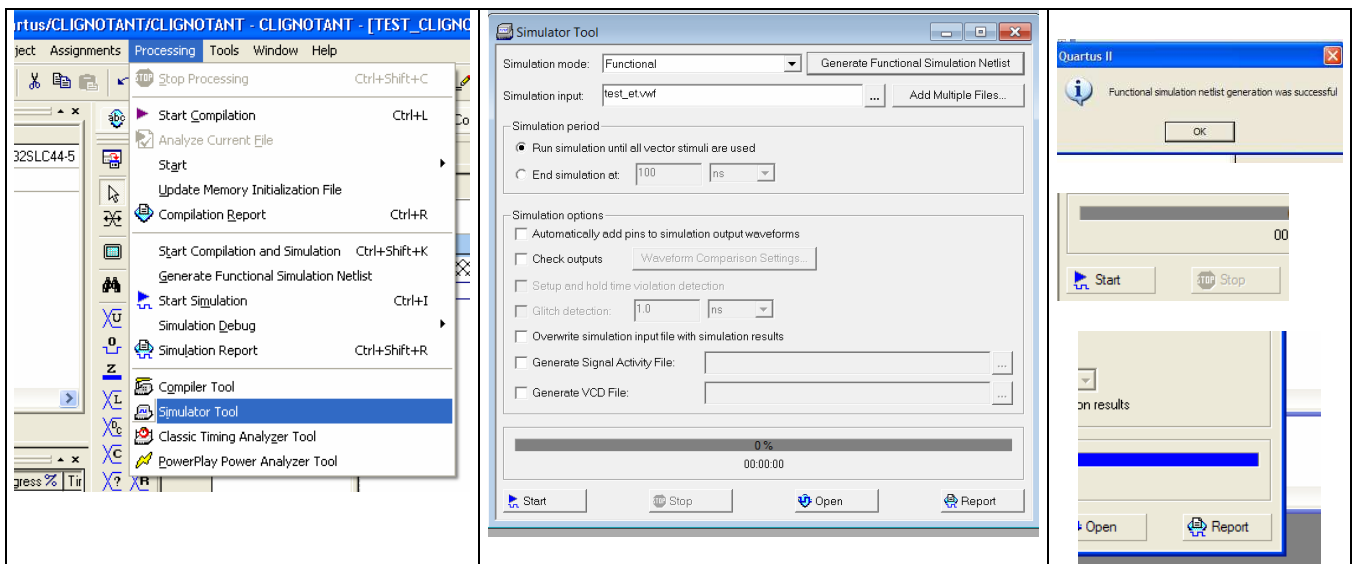


On sauve ensuite le fichier avec un nom évocateur (test_et par exemple) avec une extension « .vwf » pour « Vector Waveform File ».

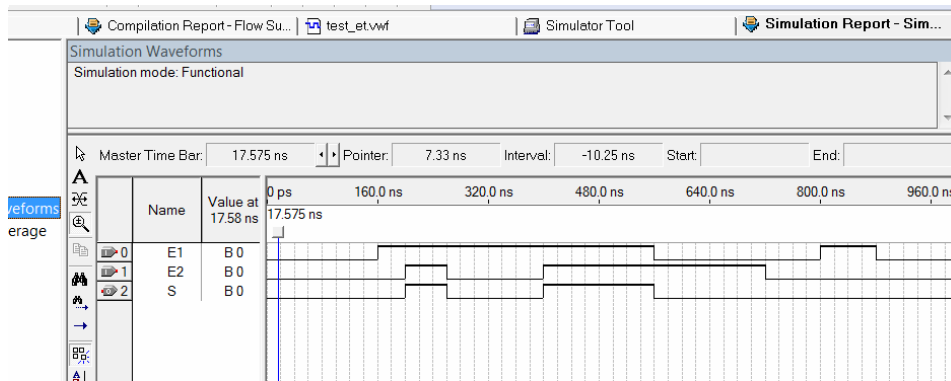
2.4.2 Lancer une simulation fonctionnelle

Ouvrir pour cela le simulateur :

- valider l'option « **Fonctional** » ;
- déclarer le fichier précédent « **test_ET** » comme entrée de simulation ;
- choisir les options comme indiqué ci-après ;
- générer la liste de connexions (netlist) nécessaire à la simulation par « **Generate Functional Simulation Netlist** ».
- lancer la simulation par « **Start** » ;
- afficher le résultat par « **Report** ».



Attention : dans certaines versions précédentes de Quartus, le sous menu « Simulateur Tools » se trouve dans le menu « Tools ».



Vérifier que la simulation correspond bien au résultat attendu.

Observer les différentes fenêtres qui se sont ouvertes dans l'espace de travail ; noter que le chronogramme résultat de la simulation n'est pas modifiable (pour ajouter une entrée par exemple), seul le fichier .vwf de description l'est.

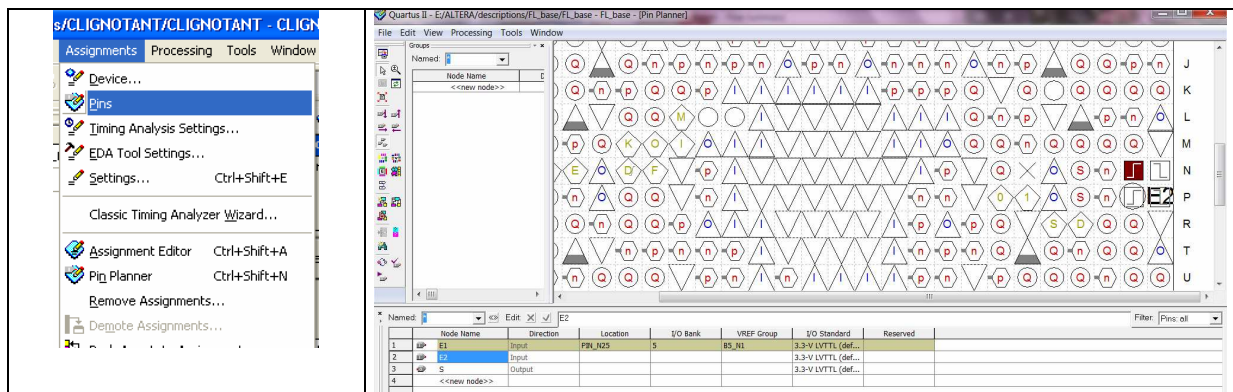
2.5 Assigner les entrées sorties

Avant de lancer la compilation du circuit, il est nécessaire d'affecter les entrées sorties aux numéros de broches de notre circuit cible, de sorte à être compatible avec la carte de développement (consulter l'annexe).

Les deux entrées seront affectées aux interrupteurs à glissière **SW0** et **SW2** (bornes **PIN_N25**, **PIN_P25**), et la sortie à la Del **LEDG8** entre les afficheurs (borne **PIN_Y12**).

Ouvrir l'**éditeur d'assignement** de broche, zoomer la zone qui nous intéresse sur le dessin du circuit cible, sélectionner un nom d'entrée sortie dans la fenêtre du bas, cliquer sur le nom, et faire glisser, à partir de la case « **Node Name** », sur le numéro de la broche correspondant. Répéter l'opération pour chaque broche.

Remarque : l'opération de « glisser déposer » étant un peu aléatoire, commencer le « glisser » par une translation horizontale au sein de la case « Node Name » (le pointeur de la souris doit prendre la forme d'un cercle barré).



Il est aussi possible de faire un double clic dans la case « **Location** » pour ouvrir une fenêtre permettant de sélectionner le numéro de broche désiré.

On peut vérifier l'affectation des broches soit dans le tableau de la fenêtre du bas, soit en passant la souris sur une broche sur le dessin du circuit.

Pour supprimer une affectation, sélectionner la broche sur le circuit et appuyer sur « **Suppr** » au clavier.

Remarques : on consultera les annexes et la documentation associées à la carte pour connaître le numéro des bornes du circuit affectées aux entrées sorties.

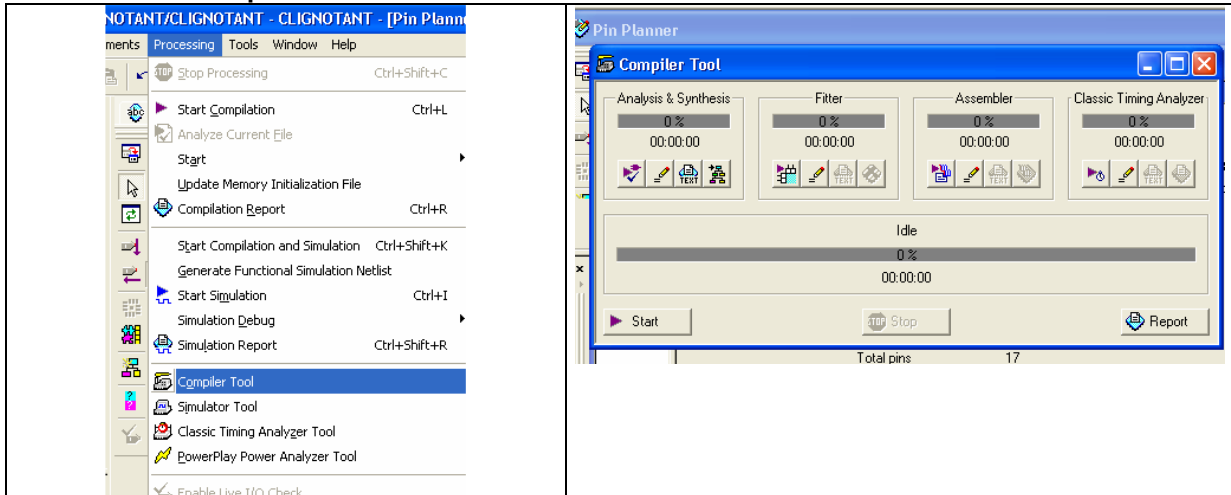
Lorsque l'on souhaite maintenir une sortie au NL1, sélectionner la broche et accéder au sous menu « **Reserve** » dans le menu contextuel (**clic droit**) puis sélectionner l'option « **As output driving Vcc** ».

Les sorties non utilisées du FPGA sont mise par défaut au niveau logique 0, mais cette option peut être modifiée par « **Assignements** » « **Settings...** » « **Device** » « **Devices and Pin Options...** » puis « **Unused Pins** » :

2.6 Compiler le projet

Lors de cette opération, le logiciel va créer les fichiers nécessaires à la programmation en définissant l'emplacement de chaque fonction logique au sein du circuit cible.

Ouvrir le compilateur et lancer la compilation (cette opération peut aussi se faire en une fois avec l'icône « **Start Compilation** »).



Attention : dans certaines versions précédentes de Quartus, le sous menu « **Compilateur Tools** » se trouve dans le menu « **Tools** ».

Remarque : le compilateur et le simulateur ayant été ouverts, ils apparaissent comme onglet dans la fenêtre de travail, il suffit maintenant de cliquer dessus pour les faire réapparaître.

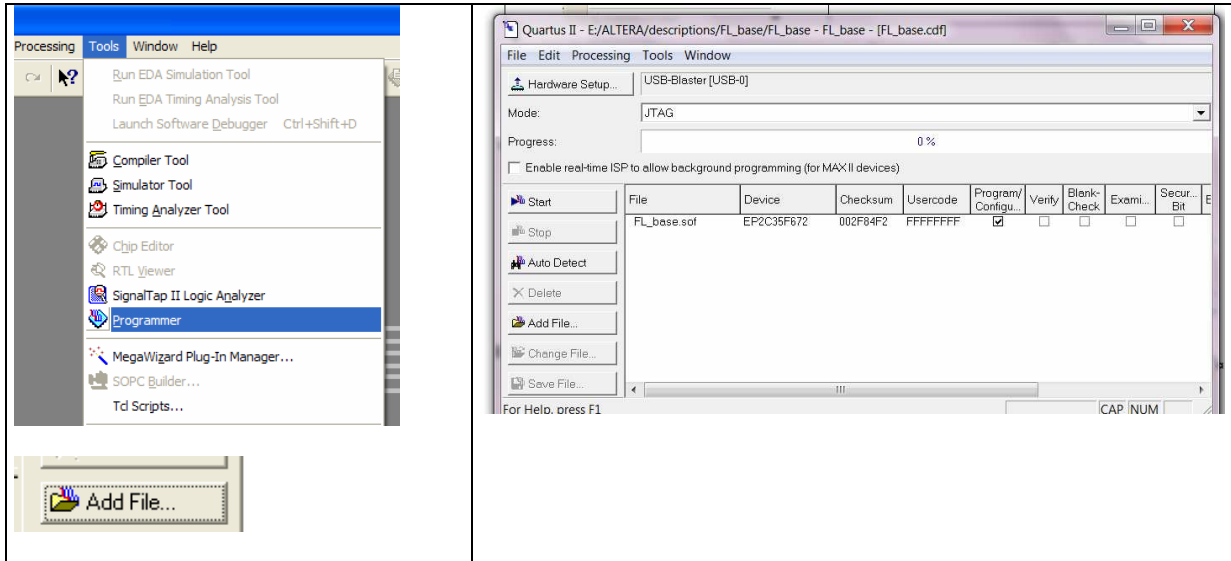
2.7 Programmer le circuit

La carte DE2 doit maintenant être :

- alimentée ;
- sous tension (bouton rouge) ;
- en position Run (bouton à glissière près de l'afficheur LCD) ;
- reliée au port USB du PC par son entrée BLASTER (à côté de la borne d'alimentation), les pilotes étant installés (voir annexes).

Ouvrir la fenêtre du programmeur et vérifier que :

- la liaison USB Blaster est reconnue (sinon se reporter à l'annexe 3) ;
- le fichier FL_base.sof de notre projet est présent (sinon l'ajouter avec « **Add file ...** ») ;
- la case « **Program/Config** » est cochée, puis lancer la programmation par « **Start** » :



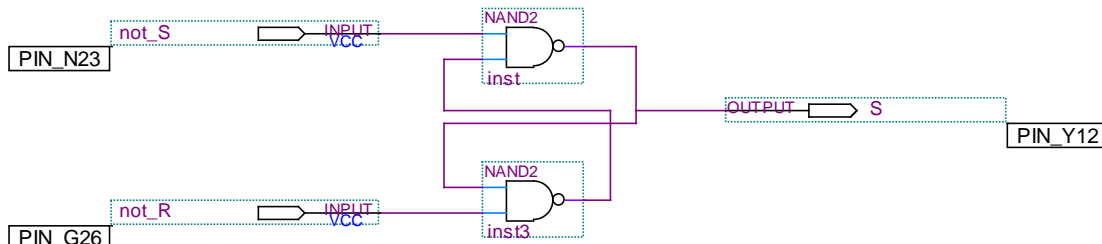
Vérifier sur la carte le fonctionnement sur la carte DE2.

2.8 Autres fonctions de bases

On pourra éventuellement tester d'autres fonctions logiques de base (OU, OU exclusif, NAND etc..) par la même méthode.

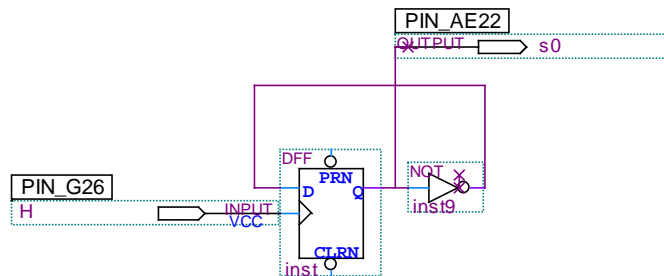
Par soucis de simplicité, on modifiera uniquement la feuille schéma en remplaçant la porte ET par une autre porte.

Tester ensuite une bascule $\bar{R} \bar{S}$ avec le schéma suivant :

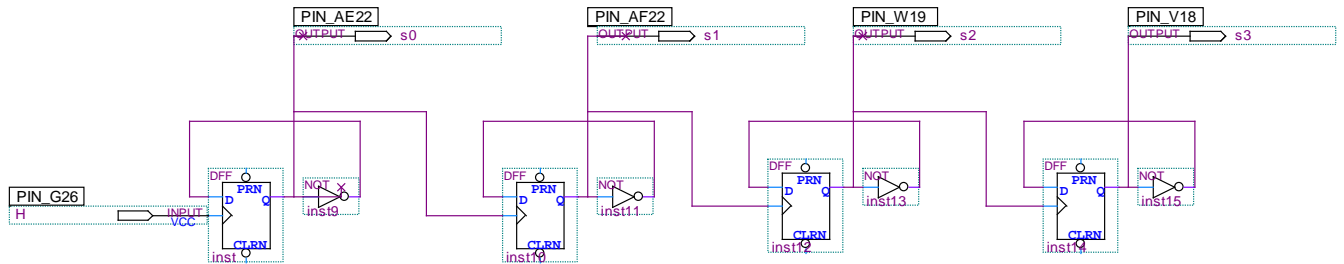


On utilisera en entrée les boutons poussoirs **KEY0**, **KEY1** (bornes **PIN_G26**, **PIN_P23**) qui sont imposent un niveau logique 0 lorsqu'on appui dessus, transformant ainsi notre bascule $\bar{R} \bar{S}$ en une bascule RS.

Tester ensuite un diviseur de fréquence par 2, réalisé à l'aide d'une bascule D (composant DFF de la bibliothèque « logic »)



Tester maintenant un décompteur binaire, commandant les leds **LEDG0**, **LEDG1**, **LEDG2**, **LEDG3** (bornes **PIN_AE22**, **PIN_AF22**, **PIN_W19** et **PIN_V18**).

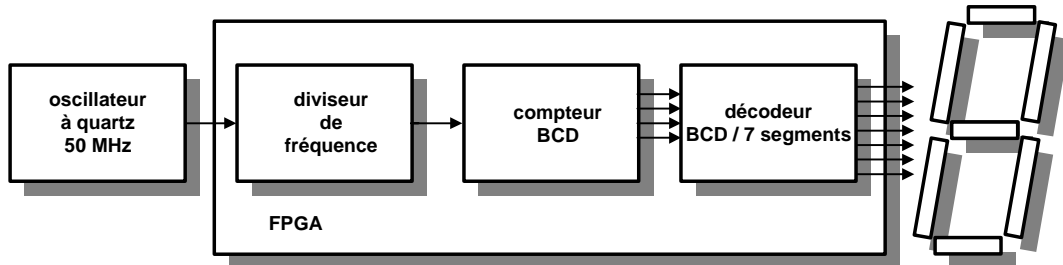


Quel est le poids binaire de la sortie s_x ?
 Quel rapport de fréquence existe-t-il entre la sortie H et la sortie s_x ?

2.9 Synthèse d'un chronomètre

2.9.1 Structure du projet

Dans un premier temps, l'affichage de notre chronomètre se fera sur un seul afficheur, puis nous ferons évoluer notre projet vers un affichage sur deux chiffres:



La carte DE2 intègre un oscillateur à quartz de fréquence 50 MHz (borne PIN_N2). Grâce à un diviseur de structure similaire au schéma que nous venons d'implanter, nous abaisserons la fréquence à 1 Hz.

Un compteur BCD (Binaire Codé Décimal) comptera ensuite le nombre de front du signal issu du diviseur, puis un décodeur BCD / 7 segments convertira la sortie BCD du compteur en code 7 segment pour commander l'afficheur.

Pour synthétiser ces différents blocs, nous utiliserons des descriptions prédéfinies et paramétrables intégrées au sein du logiciel Quartus II, désignées sous le terme « megafonctions ».

2.9.1.1 Synthèse du diviseur

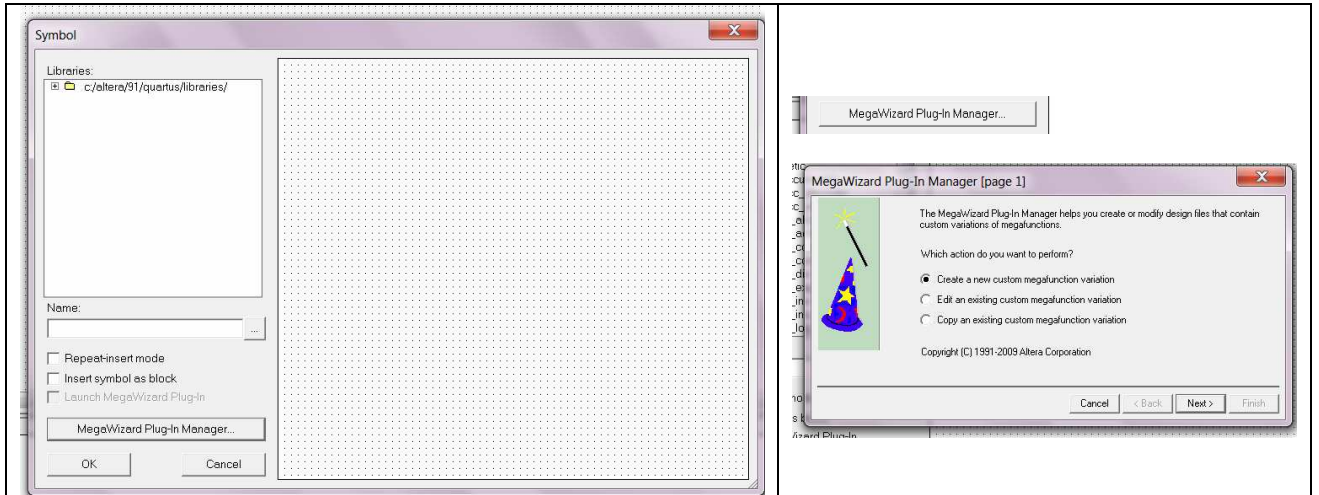
Comme nous l'avons vu précédemment, un diviseur de fréquence peut être réalisé à partir d'un compteur ou d'un décompteur.

Afin de pouvoir simuler les descriptions, nous supposons dans un premier temps que la fréquence du signal d'horloge est à 5 Hz ; le diviseur devra donc diviser par 5 pour fournir en sortie un signal de 1 Hz, son modulo (nombre d'état de sortie du compteur) est donc de 5, et il faut trois bits pour compter jusqu'à 5.

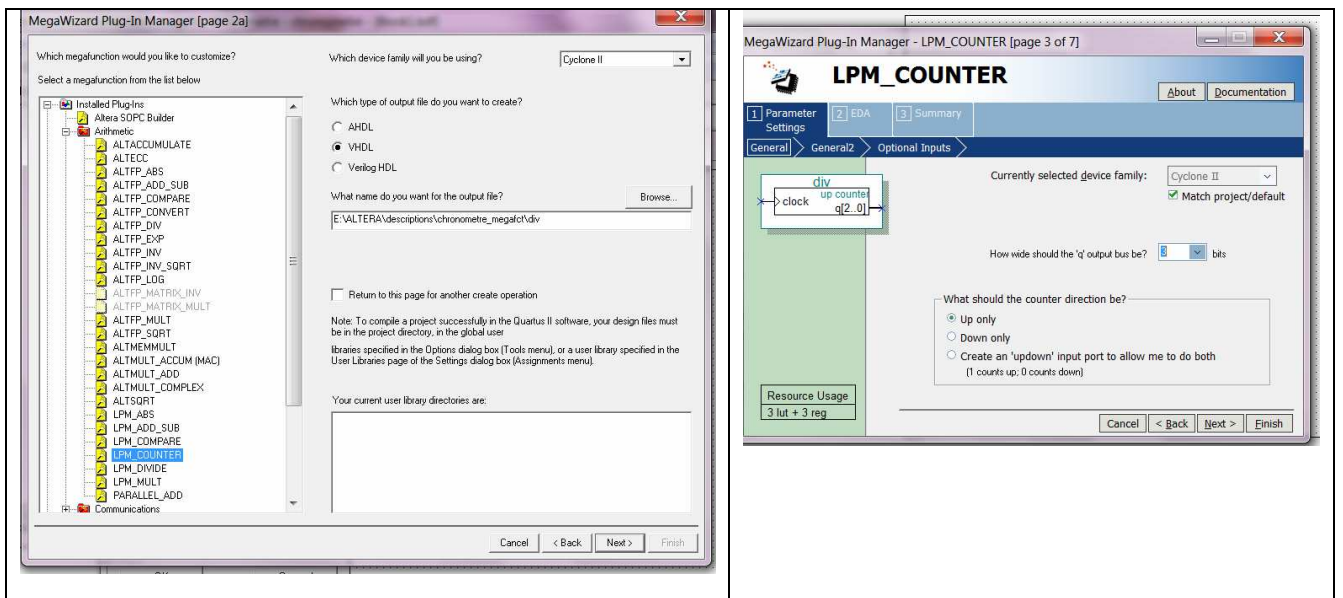
Ouvrir un nouveau projet que l'on nommera « chronometre » par exemple (sans accents), puis ouvrir dans ce projet une feuille graphique sur laquelle on placera un symbole qui sera notre diviseur.

Pour paramétrer ce dernier, on utilisera l'assistant de paramétrage :

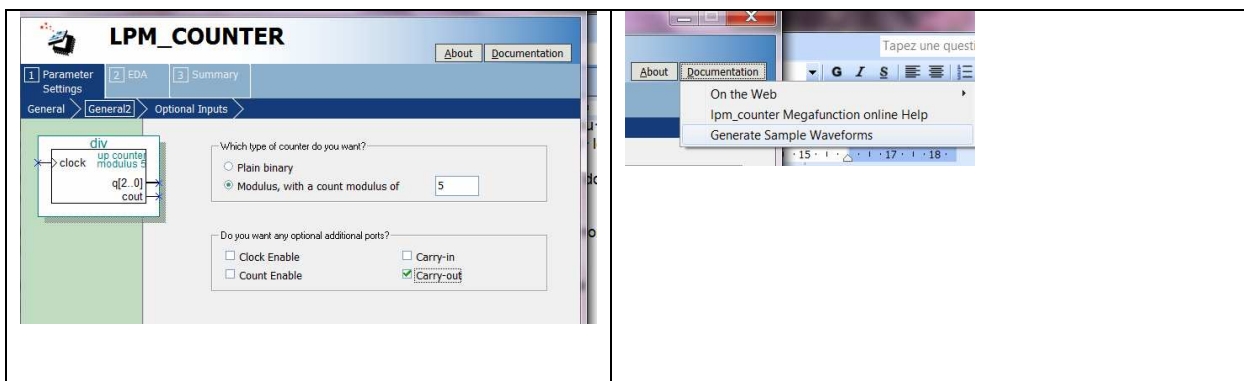
Programmer un FPGA avec Quartus II



Le compteur qui nous servira de diviseur se trouve dans la bibliothèque `c:/altera/xx/quartus/librairies/megafonctions/arithmetique/lpm_counter`.



Configurer alors les 3 bits nécessaire, pour un fonctionnement en compteur uniquement, ainsi qu'une sortie de retenue qui passera au niveau logique 1 lorsque le maximum du compteur sera atteint. Il est possible d'afficher les chronogrammes associés aux différentes entrées sorties choisies, en cliquant sur « Documentation » puis « Generate Sample Waveforms ».



Sample behavioral waveforms for design file "div.vhd"

The following waveforms show the behavior of lpm_counter megafunction for the chosen set of parameters in design "div.vhd". The design "div.vhd" is a 3 bit up modulus 5 counter.

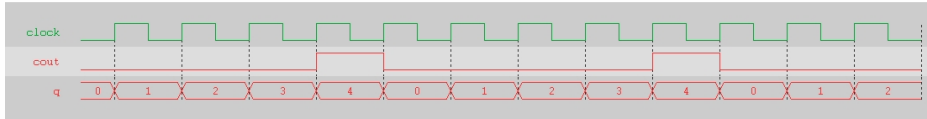
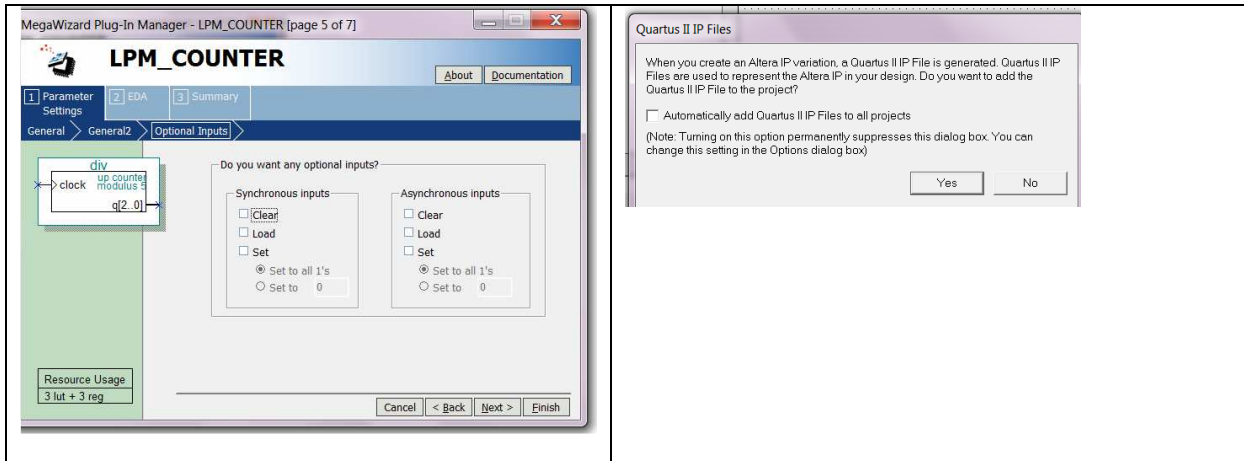


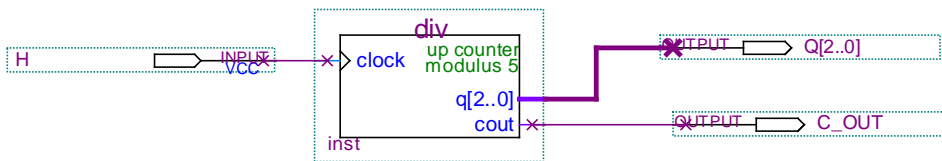
Fig. 1 : Wave showing counter operation.

The output port cout will be asserted at the completion of count sequence. The ports cin and cout are used to chain multiple counters together.

Terminer le paramétrage sans rien ajouter, en demandant d'intégrer le composant au projet.



Placer ensuite des entrées et sorties sur la feuille, effectuer une analyse et synthèse.



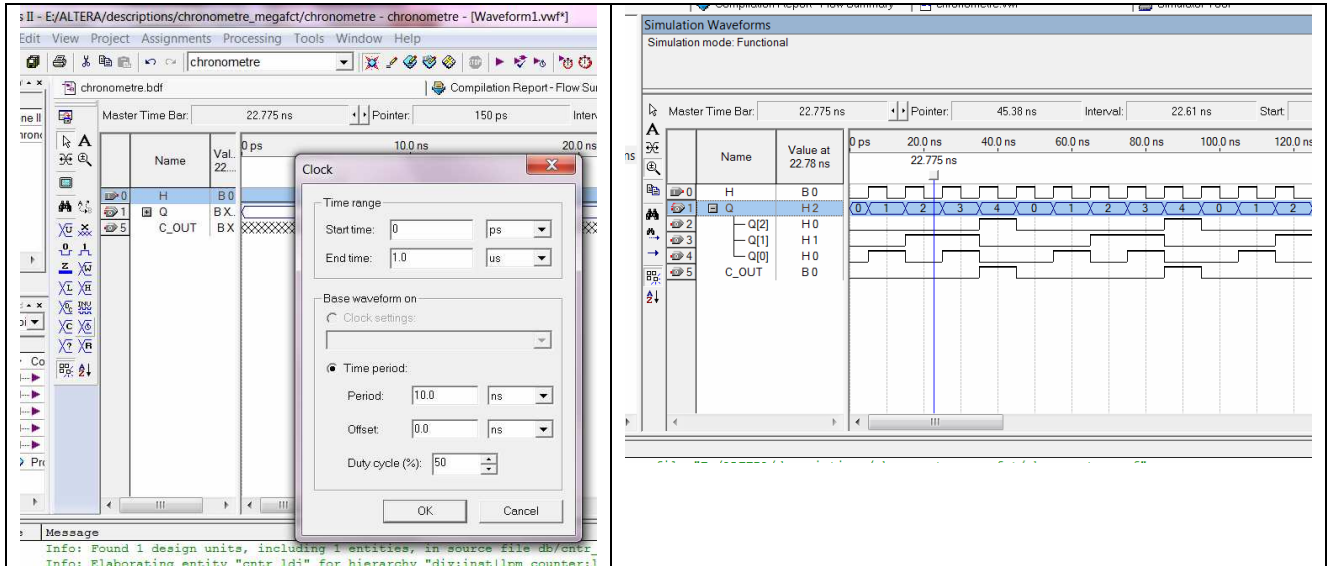
Nous allons maintenant simuler notre projet :

- ouvrir un « Waveform Vector File » ;
- placer les entrées sorties du projet ;

- imposer sur l'entrée H un signal carré avec l'outil « Overwrite Clock » ;
- sauvegarder le fichier ;
- générer la « netlist » ;
- lancer la simulation ;
- ouvrir le fichier rapport
- vérifier que le résultat est bien celui attendu.

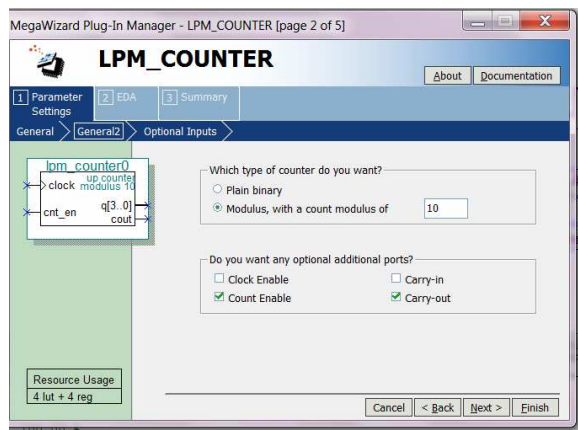
Remarques :

- il est possible de choisir le format d'affichage (radix) du bus par un double clic sur la ligne du bus ;
- il est possible de « déplier » et « replier le bus » en cliquant sur la croix devant le nom du bus.



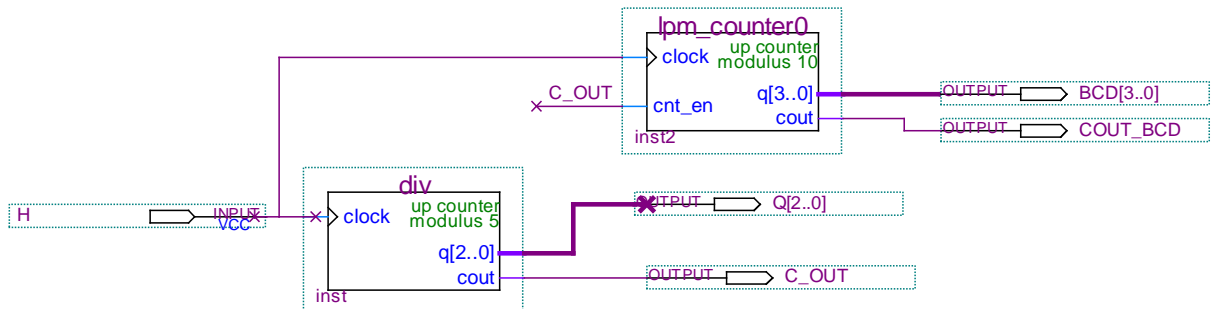
2.9.1.2 Synthèse du compteur BCD

Nous utiliserons le même composant que précédemment, mais avec 4 bits en sortie et un modulo de 10 (compteur BCD), une entrée d'autorisation de comptage, une sortie de retenue qui nous permettra par la suite un affichage sur plusieurs digits :



Insérer le compteur comme indiqué sur la figure suivante :

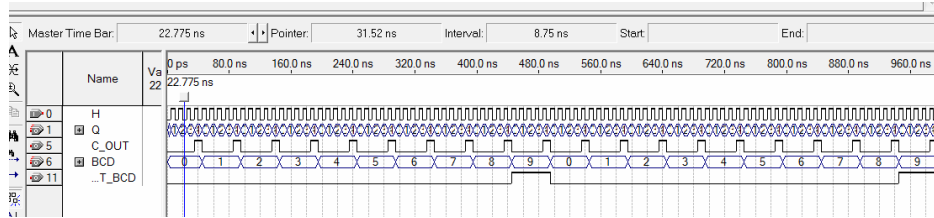
Remarque : afin de ne pas surcharger le schéma, l'entrée d'autorisation de comptage du compteur BCD est relié à un fil appelé C_OUT ; le logiciel interprète alors que cette entrée est reliée à tout autre connexion de même nom. Pour donner un nom à une connexion, il suffit de la sélectionner et d'écrire le nom désiré.



On peut remarquer que l'horloge est commune aux 2 composants, ce qui permet d'éviter les aléas de fonctionnement dus aux temps de propagation de cette horloge. On dit alors que le système est synchrone, contrairement au décompteur que nous avons réalisé avec des bascules D, qui lui était asynchrone.

L'électronique numérique moderne est systématiquement synchrone, et un logiciel comme Quartus II produirait rapidement un fonctionnement erroné du FPGA avec un système asynchrone dès que la complexité augmente.

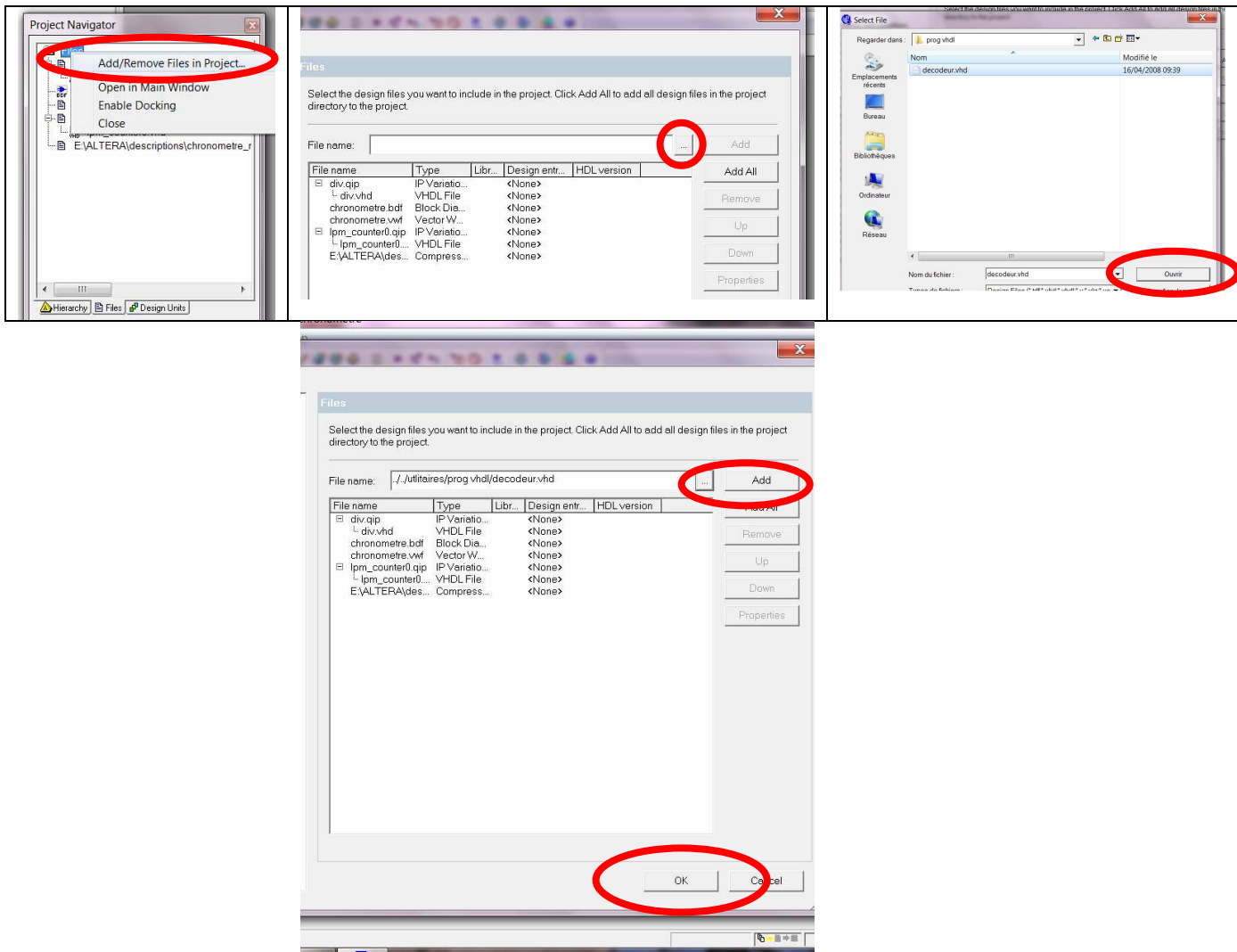
Modifier le vecteur de simulation pour introduire les nouvelles sorties et simuler le schéma.



2.9.1.3 Synthèse du décodeur BCD 7 segments

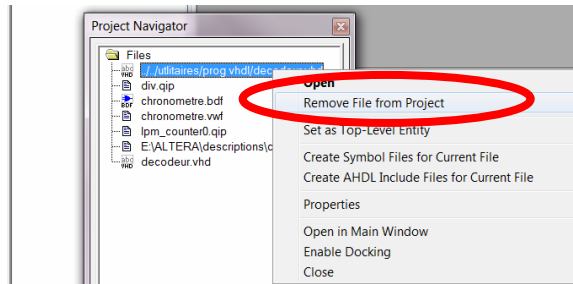
Cette fonction, purement combinatoire, n'existe pas sous forme de megafonctions, une alternative est donc de la décrire avec un langage adapté aux circuits programmables comme le VHDL (Very High scale integrated circuits Description Language). Cette description est déjà réalisée dans le fichier « decodeur.vhd » :

- ajouter ce fichier à votre projet par le menu contextuel de l'onglet « Files » du navigateur de projet ;

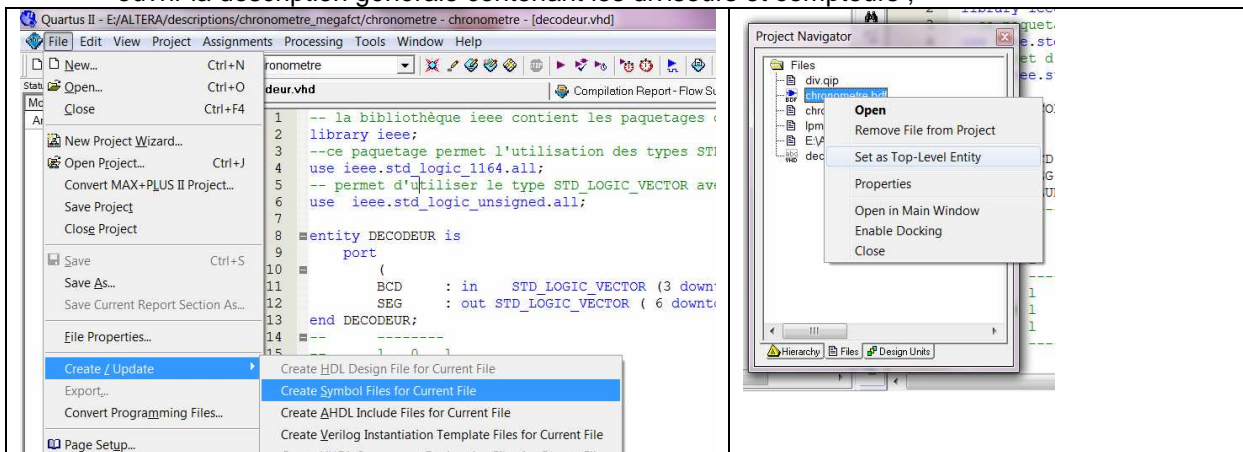


- ouvrir le fichier ;
- le sauvegarder au sein de votre répertoire de travail, sans en changer le nom ;
- supprimer du projet la version d'origine

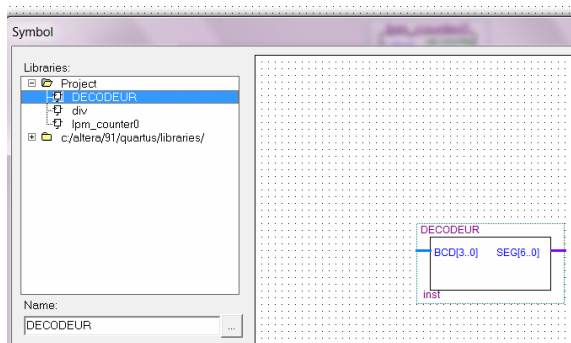
Programmer un FPGA avec Quartus II



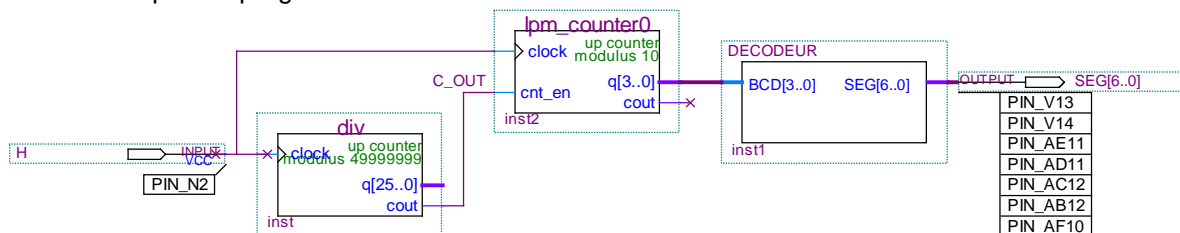
- le placer en haut de la hiérarchie (Ctrl Shift J la fenêtre correspondant à la description étant active)
- lancer une analyse et synthèse ;
- créer un symbole graphique pour cette description ;
- ouvrir la description générale contenant les diviseurs et compteurs ;



- insérer dans la description générale le décodeur qui est maintenant un symbole du répertoire projet ;



- modifier le nombre de bits du diviseur et son modulo afin de l'adapter à l'horloge de 50 MHz ;
- supprimer les sorties inutiles, placer le bus de commande de l'afficheur ;
- effectuer une analyse et synthèse (avec un Ctrl Shift J d'abord pour être sûr que le projet sur lequel vous travaillez est bien en haut de la hiérarchie) ;
- affecter le broches correspondant à l'horloge et à l'afficheur (voir annexe) ;
- compiler et programmer le circuit



Vérifier que le fonctionnement est bien celui attendu.

Modifier maintenant la description pour commander deux afficheurs ; le système devra être synchrone, il faudra donc que tous les compteurs aient la même horloge, le comptage n'étant autorisé par l'entrée « cnt_en » que si le compteur de la décade précédente a sa sortie « cout » au niveau logique 1. On pourra éventuellement faire des simulations (en modifiant le rapport de division du diviseur).

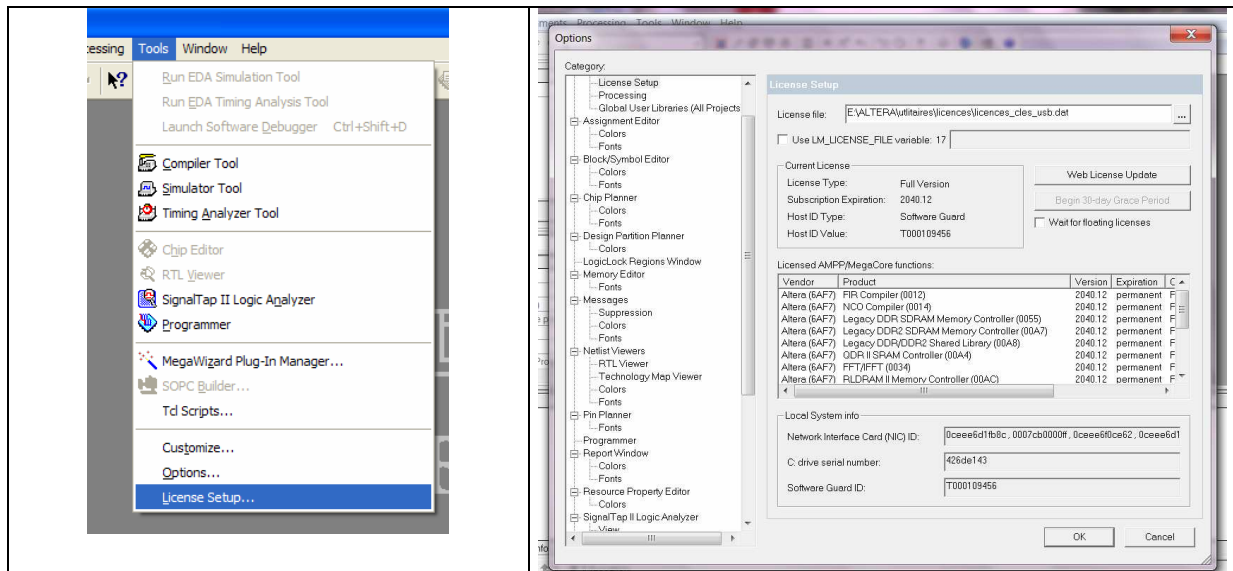
Annexe 1 : Obtenir et configurer la licence

Quartus II est disponible gratuitement sur le site d'Altera <http://www.altera.com>.

Après installation, lors de la première ouverture du logiciel (ou si la licence n'est pas valide), Quartus propose trois options :

- travailler avec une version d'évaluation pendant un mois ; il ne sera alors pas possible de programmer de circuit ;
- effectuer une demande automatique de licence sur le site internet d'Altera ; sous réserve de posséder une connexion, vous serez alors basculé vers la page de demande de licence, où après avoir complété les différents renseignements, vous recevrez un fichier « .dat » par courriel (il faut donc une adresse valide) ; la licence étant reconnue à partir de la carte réseau de votre ordinateur, celui-ci devra en intégrer une ;
- spécifier un fichier licence valide (ce qui suppose d'avoir effectué l'étape précédente).

Une fois votre fichier licence reçu par courriel (cas 2), il faut le déclarer au logiciel (cas 3). Pour cela, copier le fichier « .dat » sur votre disque dur et déclarer l'emplacement comme suit :



Remarques :

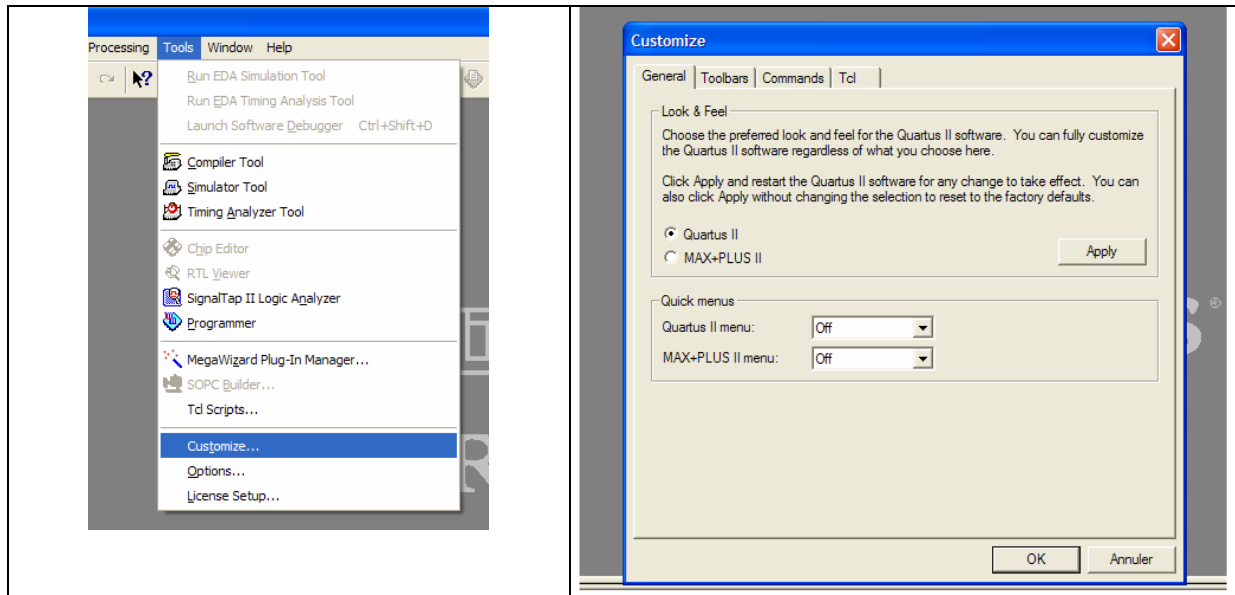
- pour obtenir la licence de « Quartus II Web Edition », un ordinateur PC équipé d'une carte réseau suffit ; le numéro (NIC ID) de cette carte vous sera demandée en cas d'inscription manuelle ou sera lu lors de la connexion en cas d'inscription automatique. La licence ne sera valable qu'associée à cette carte réseau.
- pour obtenir la licence de la version complète, une clé matérielle (associée à son numéro « Software Guard ID ») sera nécessaire ; la licence ne sera alors valide que si la clé matérielle est connectée à l'ordinateur ; la première génération de clé matérielle se connecte sur le port parallèle, tandis que la seconde génération se connecte sur un port USB, il faut alors installer un pilote disponible à l'adresse internet suivante : http://www.safenet-inc.com/support_and_downloads/download_drivers/sentinel_drivers.aspx#
- la licence est simplement un fichier texte (ouvrable avec n'importe quel éditeur) avec une extension « .dat » ; lorsque plusieurs licences sont utilisées (dans le cas où Quartus sera installé sur plusieurs PC par exemple), il est possible de les regrouper tous les fichiers licence en un seul fichier « .dat » où seront copiés les textes associés à chaque fichier élémentaire à l'aide du Bloc Note de Windows par exemple ; attention **ne pas utiliser Word** pour cette opération, ce programme introduisant des caractères spéciaux rendant le fichier illisible pour Quartus.

Annexe 2 : Configurer l'affichage

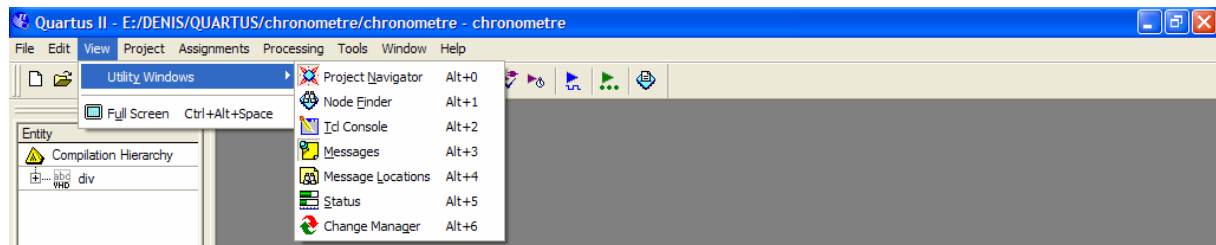
Deux interfaces graphiques sont envisageables avec Quartus II :

- l'interface « classique » ;
- l'interface rappelant les menus de max+plus II (le logiciel de génération précédente chez Altera).

Nous travaillerons ici avec la première interface, obtenue de la manière suivante :

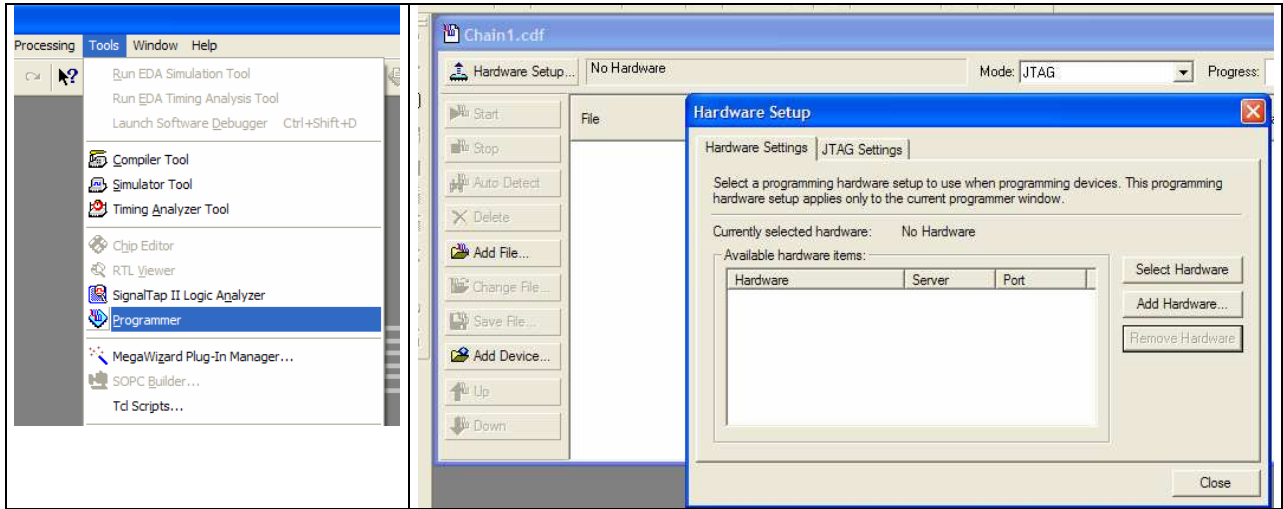


Quartus ouvre beaucoup de fenêtre, il est possible de choisir ses fenêtres par le menu suivant :

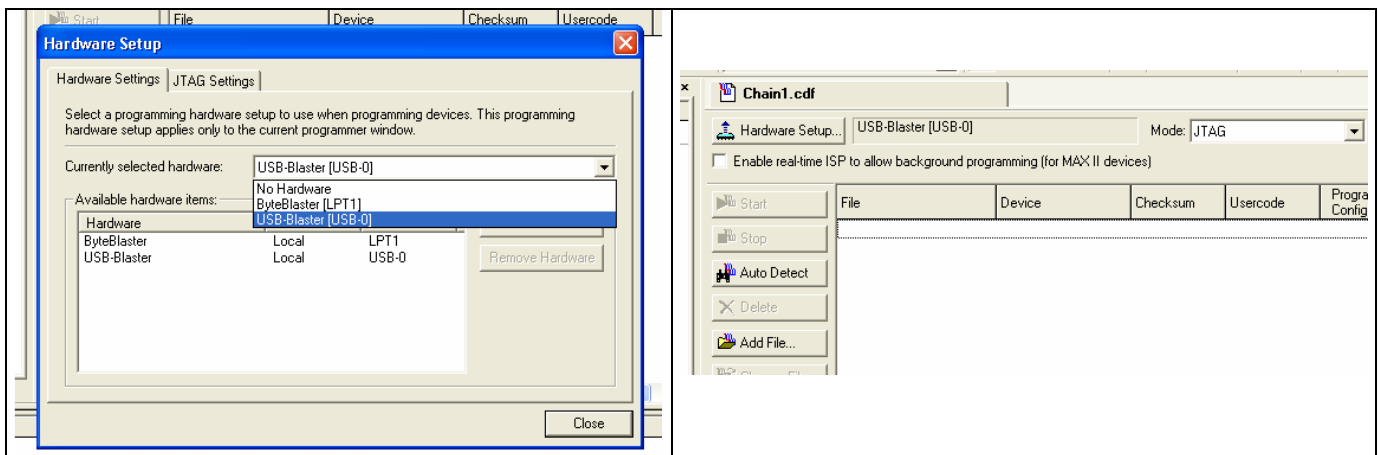


Annexe 3 : Configurer le programmeur

Ouvrir Quartus et lancer le programmeur puis cliquer sur « **Hardware Setup....** »



Sélectionner « **USB Blaster [USB-0]** » dans « **Currently selected hardware** » et fermer la fenêtre par « **Close** » (le câble USB doit être connecté et la carte sous tension):



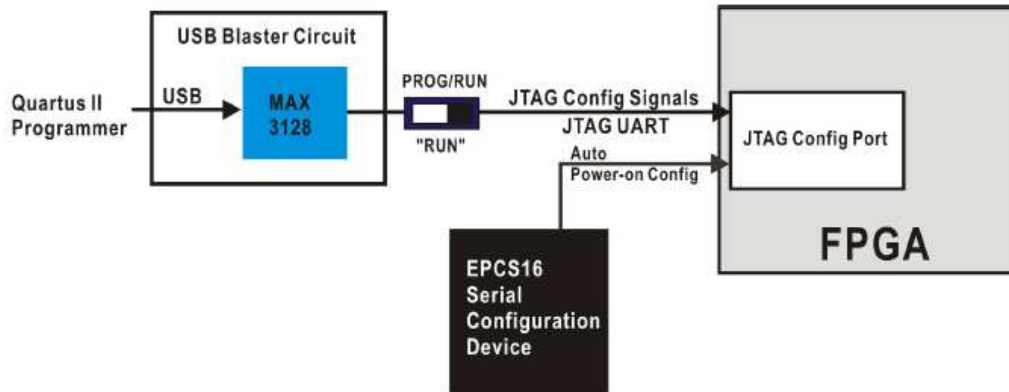
Choix du mode de configuration de la carte DE2

Le FPGA étant réalisé en technologie SRAM, les données programmées disparaissent à chaque coupure d'alimentation ; une mémoire série EEPROM EPCS16 est implantée sur la carte DE2 afin de garder le programme. Deux modes de configuration sont alors possibles à partir de Quartus.

Configuration du FPGA en mode JTAG

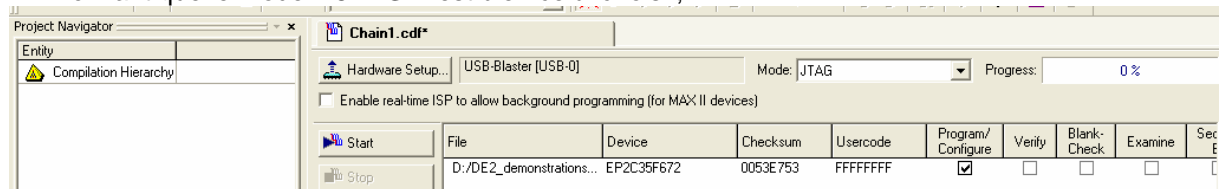
On configure directement le FPGA sans tenir compte de l'EEPROM, les données étant alors perdues à chaque coupure d'alimentation :

Programmer un FPGA avec Quartus II



Pour ce mode :

- mettre la carte sous tension, la relier au PC par le câble USB via le port « USB blaster » ;
- placer le commutateur RUN/PROG (bord gauche de la carte) en position **RUN** ;
- configurer Quartus en sélectionnant dans le programmeur le logiciel « .sof » souhaité et en vérifiant que le mode « **JTAG** » est bien celui choisi ;



- cliquer sur « **Start** », le FPGA est configuré.

Nous nous limiterons à cette utilisation de la carte.

Annexe 5 : Quelques caractéristiques succinctes de la carte de développement DE2

Les informations suivantes donnent quelques éléments d'utilisation de cette carte ; pour plus d'information on consultera la documentation du constructeur ainsi que le document « Première utilisation de la carte DE2 ».

Alimentation continue

Tension continue de 9 V 1,3 A, polarité positive au centre du connecteur.

Configuration

La carte est programmée par un câble USB Blaster reliant le PC au connecteur « USB Blaster » en haut à gauche de la carte. Le pilote logiciel de cette liaison doit être installé (voir « Première utilisation de la carte DE2 »).

Mettre la carte sous tension à l'aide du bouton rouge en haut à gauche et la placer en mode « RUN » grâce au commutateur à gauche de l'afficheur LCD (pour plus de détails voir le document « Première utilisation de la carte DE2 »).

Le FPGA est alors directement programmé avec un fichier « .sof »

Oscillateurs

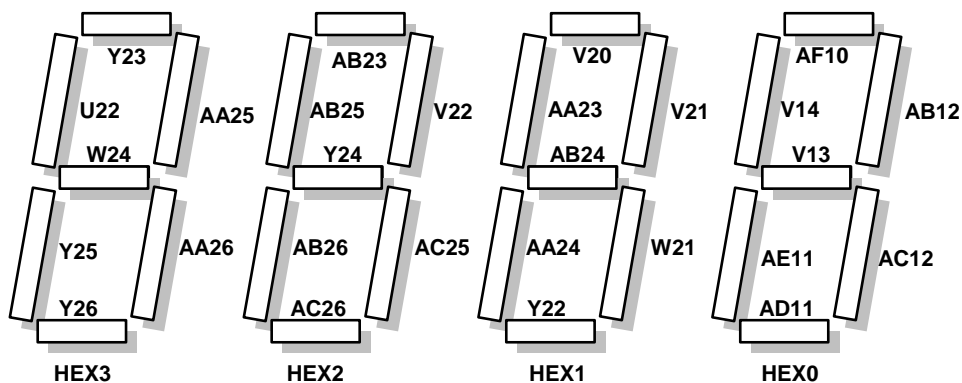
Le FPGA (un EP2C35F672C6 de la famille Cyclone II pour une carte DE2 et un EP2C70F896C6 pour une carte DE2-70) reçoit trois signaux d'horloge :

- une horloge à **50 MHz** sur la borne **PIN_N2** ;
- une horloge **externe** sur la borne **PIN_P26** à envoyer sur le connecteur **SMA** (coin inférieur droit de la carte) ;
- une horloge à **27 MHz**, via le décodeur vidéo, sur la borne **PIN_D13**. Pour que cette horloge arrive au FPGA, il est impératif que la borne RESET du décodeur vidéo, reliée à la borne **PIN_C4** du FPGA, soit au **niveau logique 1**. Cette dernière peut être activée au NL1 au moment de l'affectation des broches sous Quartus ; pour cela, sélectionner la broche et accéder au sous menu « **Reserve** » dans le menu contextuel (**clic droit**) puis sélectionner l'option « **As output driving Vcc** ».

Afficheurs et Del

La Del **LEDG8** (entre les afficheurs) est reliée à a borne **PIN_Y12**.

Les afficheurs de droite (HEX0 à HEX3) sont reliés aux bornes du FPGA comme indiqué ci-dessous :



Interrupteurs et boutons poussoirs

La carte comprend 18 interrupteurs (en bas à gauche) à glissière, relié directement au FPGA ; lorsque l'interrupteur est en position basse, il impose un niveau logique 0.

Les interrupteurs **SW0**, **SW1** et **SW2** sont respectivement reliées aux bornes **PIN_N25**, **PIN_N26** et **PIN_P25** du FPGA.

On trouvera également 4 boutons poussoirs (en bas à droite), attaquant le FPGA via un anti-rebond. Un appui sur le bouton poussoir provoque un niveau logique 0 à l'entrée du FPGA.

Les interrupteurs **KEY0**, **KEY1**, **KEY2** et **KEY3** sont respectivement reliées aux bornes **PIN_G26**, **PIN_P23**, **PIN_N23** et **PIN_W26** du FPGA.

Annexe 5 : principales extensions de fichiers

fichier descriptif du projet : **.qpf**

Fichiers de description

description graphique : **.bdf**

description vhdl : **.vhd**

description par chronogrammes (fichier de simulation) : **.wdf**

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**

Annexe 6 : programme VHDL du décodeur

```

-- la bibliothèque ieee contient les paquetages dont la déclaration suit
library ieee;
--ce paquetage permet l'utilisation des types STD_LOGIC et STD_LOGIC_VECTOR
use ieee.std_logic_1164.all;
-- permet d'utiliser le type STD_LOGIC_VECTOR avec des entiers
use ieee.std_logic_unsigned.all;

entity DECODEUR is
    port (BCD : in STD_LOGIC_VECTOR (3 downto 0);
          SEG : out STD_LOGIC_VECTOR ( 6 downto 0) );
end DECODEUR;

--
--      -----
--      |      0      |
--      | 5          | 1
--      |              |
--      |      -----
--      |      6      |
--      | 4          | 2
--      |              |
--      |      -----
--      |      3      |
--
architecture ARCH_DEC of DECODEUR is
begin
-- attention les leds s'allument pour une sortie à 0
    SEG <= "1000000" when BCD=0 else
           "1111001" when BCD=1 else
           "0100100" when BCD=2 else
           "0110000" when BCD=3 else
           "0011001" when BCD=4 else
           "0010010" when BCD=5 else
           "0000010" when BCD=6 else
           "1111000" when BCD=7 else
           "0000000" when BCD=8 else
           "0010000" when BCD=9 else
           "1111111" ;
end ARCH_DEC ;
    
```

Zone déclarant l'utilisation de ressources externes

les commentaires, précédés de deux tirets

l'entité, décrivant une boîte noire avec des entrées sorties d'un type particulier

Zone de commentaire décrivant comment doit être connecté le bus SEG à l'afficheur

l'architecture

Affectation des valeurs au bus SEG, afin d'allumer le bon segment, en fonction des valeurs du bus BCD