Objectifs : Le flot de synthèse VHDL

I Introduction

L'impact de l'écriture comportementale VHDL sur l'inférence des blocs se vérifié à partir du schéma RTL donnant une vision technologiquement indépendante de la description VHDL (schéma RTL). L'obtention du schéma technologique depuis ce schéma RTL s'obtient ensuite après énoncé de directives d'optimisations permettant de guider l'outil vers un circuit optimisé selon certains critères. Les performances de l'implémentation peuvent alors être analysées. Enfin, le processus de synthèse s'achève par le floorplanning du circuit conduisant à l'implémentation physique du circuit et à sa programmation pour le test du circuit. Cette étape implique l'utilisation du logiciel spécifique à la technologie utilisée (Quartus dans notre cas) et la définition de contraintes pour l'affectation des pattes entre autres.

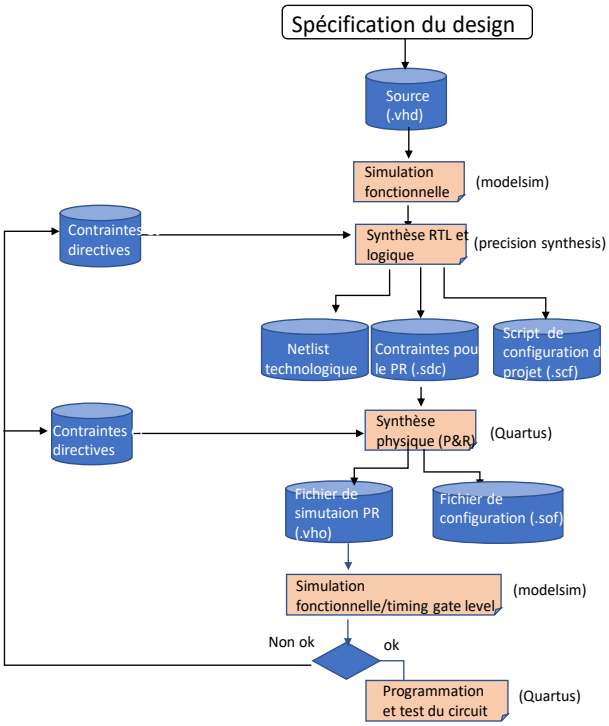


Figure 1 : Flot de synthèse d'une description VHDL

Dans cette manipulation, on s'intéresse au flot de synthèse menant d'une description comportementale VHDL d'un circuit le bloc display du modem) jusqu'à son implémentation sur composant FPGA. On vous propose d'illustrer le processus de synthèse du FPGA en 5 parties :

1. Simulation fonctionnelle RTL du bloc (etape similaire aux TP vhdl tronc commun)
2. Synthèse et optimisation logique et placement routage à partir de menus
3. Synthèse et optimisation logique et placement routage à partir d'un script de commandes
4. Simulation post routage sous modelsim
5. Test du circuit sur la carte DE2

II Le bloc display

On rappelle que ce bloc (cf Figure 2) a pour rôle d'enregistrer et de visualiser le mode de configuration du modem (entrée *mode)*, à l'aide d'afficheurs 7 segments. Tant que le signal en entrée *dtr* vaut 1, le mode est mémorisé sur *Mm* et affiché en sortie sur 2 afficheurs permettent d'indiquer le mode de transmission, bidirectionnel (FD) ou à l'alternat (HD), les 4 autres afficheurs indiquent le débit de transmission en bps (400,800 ou 1200).

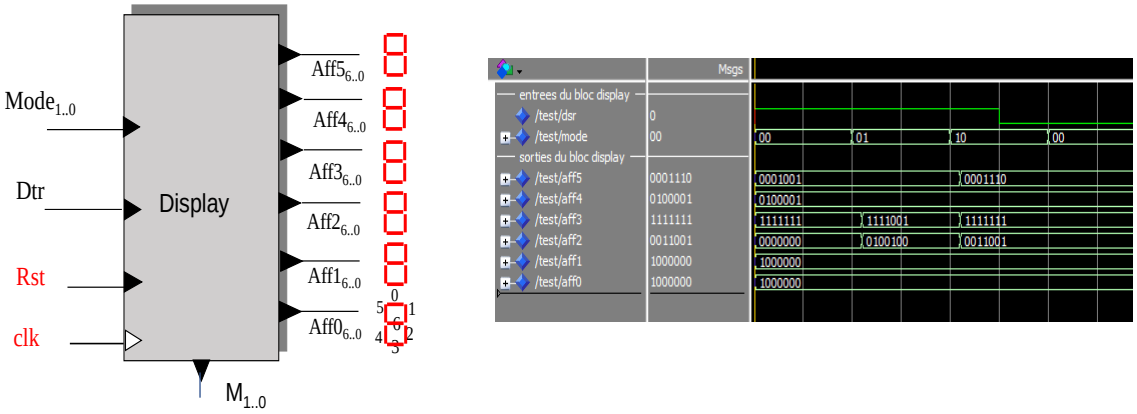


Figure 2:Symbole et spécification du bloc display

La figure 2 montre un algorithme synchrone (ASM) permettant de définir le comportement du bloc au niveau horloge du circuit (signaux *clk* et *rst* en rouge) et obtenir ainsi une description synthétisable. La variable *m* permet de conserver en mémoire la dernière valeur de l'entrée *mode* échantillonnée sur l'occurrence d'un front d'horloge *clk*. (transfert indiqué par ← dans l'ASM). Le transfert est conditionnel et ne s'applique que si *dtr* vaut 1. La sortie est identique à cette variable. La valeur des segments des afficheurs est quant à elle déterminée à tout instant par *m* et donc réalisés de manière asynchrone (transfert indiqué par =). La figure montre également une description structurelle qui peut être dérivée de cette description en allouant les différentes composants (registres, opérateurs arithmétiques et logiques, mux, etc) pour matérialiser cet algorithme. Comme le montre le schéma RTL partiel du circuit (figure 2), une partie du circuit est séquentielle, l'autre est combinatoire (indépendante de toute horloge). La partie séquentielle (en bleu) se limite à un registre avec une entrée de contrôle du chargement reliée à l'entrée *Dtr*. La partie combinatoire (en jaune) est formée d'opérateurs logiques déterminant les valeurs affectées aux segments en fonction de *M*. On observe que certains segments ont une valeur constante, indépendante de *M* (tous les segments des afficheurs AFF1 et AFF0 par exemple). On notera qu'au niveau RTL, on parle d'opérateurs booléens et non de portes car l'expression est indépendante de la technologie ciblée.

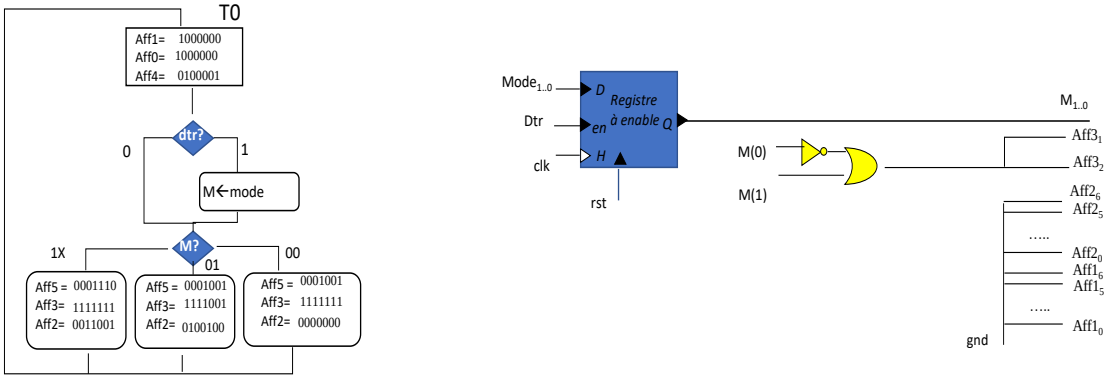


Figure 3 : ASM et schéma RTL partiel du bloc display

Ce type de solution architecturale peut s'obtenir à partir d'une description VHDL décrivant les transferts de registres pour la variable interne et les 42 équations ou valeurs à affecter aux sorties de manière asynchrone L'écriture est directe à partir du schéma et consiste à décrire l'expression de chaque signal ou groupe de signaux à partir d'instructions d'affectation concurrentes. Une alternative consiste à utiliser une écriture algorithmique, plus compacte et plus proche alors de la spécification fournie par l'Algorithmic State Machine. Des instructions *if,case,..* et donc des *process* sont alors utilisés avec l'obligation de respecter certaines règles d'écriture pour obtenir une allocation (inférence) correcte des blocs du schéma par l'outil. Dans le cas du bloc *display*, une des règles consiste à distinguer la partie séquentielle de la partie combinatoire.

III. Préparation *(A faire viser par l'enseignant au début du TP)*

A. Dans un fichier *display.vhd*, définissez l'interface du circuit à l'aide d'une entité *display*. Les signaux d'entrée et de sortie seront de type *std_logic* ou *std_logic_vector*.

B. Définissez ensuite une architecture comportementale *beh* sur la base de l'ASM figure 1, avec :

- un process synchrone à *clk* décrivant le transfert de registre à réaliser pour *mm*, sera forcé à 00 sur reset asynchrone vrai à l'état bas.
- un process combinatoire décrivant les combinaisons de sortie des afficheurs en fonction de *mm* en utilisant une instruction *case.*

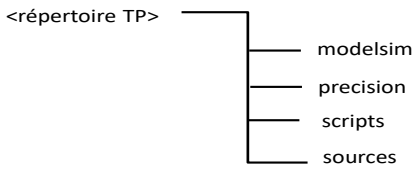
C. Ecrivez un testbench approprié pour valider la description. La couverture de test devra être suffisante pour s'assurer de la conformité du modèle RTL avec la spécification. On prendra *20 ns* de période d'horloge.

D. Ecrivez un script display.do (ou .tcl) pour la simulation RTL de ce bloc sous modelsim. On séparera clairement les signaux d'entrée, locaux et de sortie avec des commandes tel que

```
add wave -noupdate -divider {sorties du bloc display}
```

IV Travail en séance

Créez un répertoire *TP1* Attention à ce que le chemin ne contienne pas d'espaces ou caractères spéciaux. Créez l'arborescence ci-dessous (à faire pour chaque séance de TP)



Placez vos fichiers vhdl dans le répertoire *sources.*

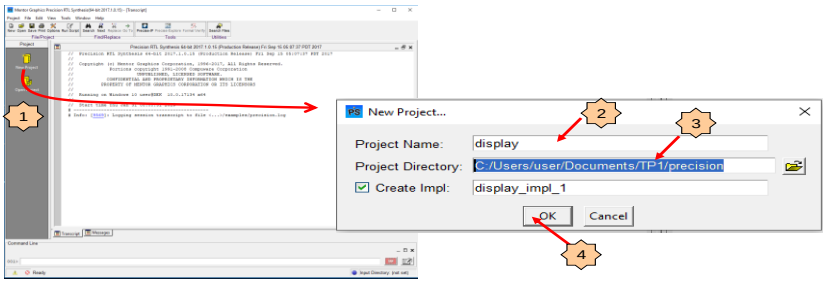
1ère partie : Validation fonctionnelle du modèle RTL

Lancez *modelsim.* Dans le sous-répertoire *modelsim,* testez et validez les architectures du bloc display sous ModelSim., avec une couverture de test suffisante. Effectuez une capture d'écran du chronogramme et commentez soigneusement.

2ème partie : Synthèse de la description VHDL par menus intéractifs

Lancez *précision synthesis RTL* 17.1 depuis le bureau.

A. Création du projet




```
> new_project -name display -folder C:/Users/user/Documents/TP1/precision -createimpl_name display_impl_1
```

Figure 4: Création du projet

A partir des menus intéractifs,

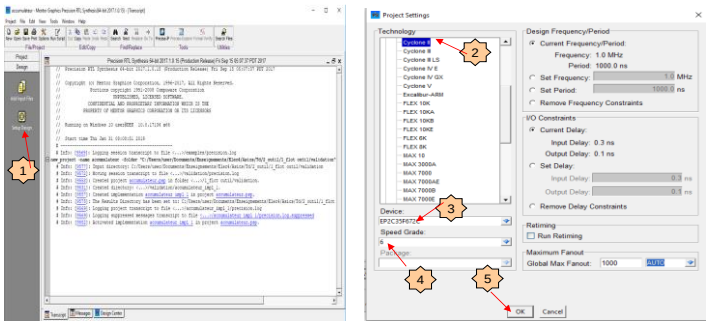
1. Cliquez sur “New Project”
2. Saisir le nom du projet
3. Saisir le nom du répertoire de projet en adaptant le chemin si nécessaire (sans espaces ni caractères spéciaux.
4. Cliquez sur le bouton OK

La ligne de commande vous permet de remplir la même fonction depuis la fenêtre *command line* ou depuis un fichier script de commandes.

B. Settings

B.1 Sélection de la technologie

Precision Synthesis dispose d’algorithmes d’optimisation dépendant de la technologie. Ainsi, la première étape est de sélectionner et charger la librairie technologique.




```
setup_design -manufacturer Altera -family "Cyclone II" -part EP2C35F672C -speed 6
```

Figure 5 : Sélection du circuit cible.

Ainsi, « Precision Synthesis » fournit un ensemble de librairies de FPGA et d’ASICs des vendeurs comme Actel, Altera, Atmel, Lucent, Xilinx, etc. Ces librairies contiennent des informations concernant les cellules (nom, ports, fonctions, temps, broche) et les paramètres globaux (tables de

routage, charge, température, tension). Dans notre cas, nous utiliserons le composant *cyclone* de chez Altera. A partir des menus intéractifs,

1. Cliquez sur « Setup Design »
2. Sélectionner le circuit cible (Technology → Altera → cyclone II)
3. Sélectionner le circuit (EP2C35F672C)
4. Sélectionner la vitesse (6)
5. Cliquez sur le bouton OK

B.2. Format des fichiers de sortie

L'écriture des fichiers résultats de la synthèse s'effectue dans le répertoire du projet « accumulateur_impl_1 » lorsque la sauvegarde du projet ou de l'implémentation est réalisée (commande save_projet ou save_impl). Avant la sauvegarde, les fichiers générés sont dans le répertoire temporaire « accumulateur_temp_1 ». L'ensemble des fichiers généré comporte une netlist (.edf , .vhd, .vo) selon le format souhaité (EDIF,VHDL,Verilog) sur la base des composants de la bibliothèque technologique sélectionnée et fichier de contrainte liée au constructeur. D'autres fichiers de résultat de la synthèse peuvent également être proposés en fonction de la cible technologique. A partir des menus intéractifs,

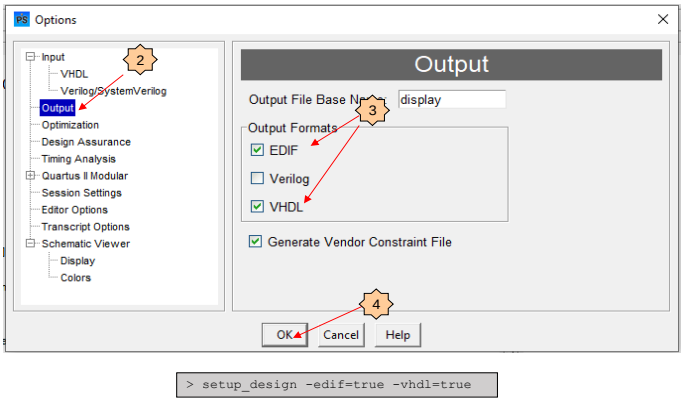


Figure 6 : Sélection du circuit cible.

1. Cliquez sur *options* →Output.
2. Sélectionner le format du fichier de sortie VHDL. Le format EDIF est coché par défaut.
3. Cliquer sur le bouton OK.

B.3 Options pour le post routage

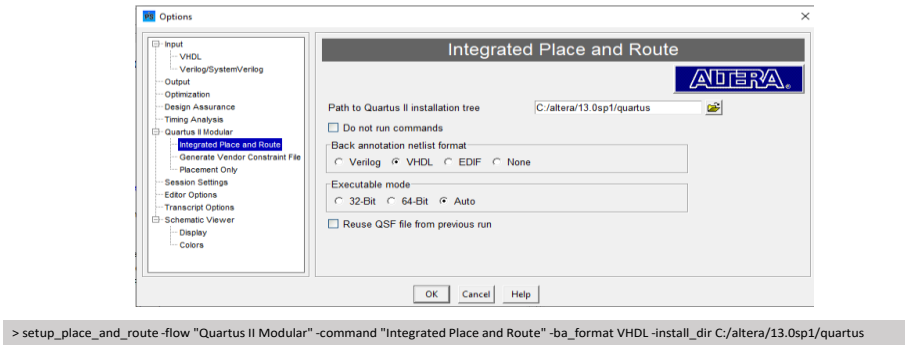


Figure 7: Format des fichiers pour le post routage

C. Compilation

L'étape de compilation permet la génération de l'architecture RTL correspondant à la description d'entrée comportementale. La synthèse RTL est effectuée sur la base de portes génériques, opérateurs arithmétiques et relationnels (+,*,…). Ces opérateurs sont vus comme des boites noires et seront ultérieurement remplacés par une solution dépendant de la technologie. Pour obtenir ce schéma,

1. Insérez la description *beh* dans le projet.
2. Cliquez sur l'icône « compile » permettant de générer le schéma RTL,
3. Double cliquez sur « RTL Schematic » (panneau design center) ou sur le bouton view RTL schematic dans l'onglet *schematics* à gauche pour faire apparaitre le schéma.

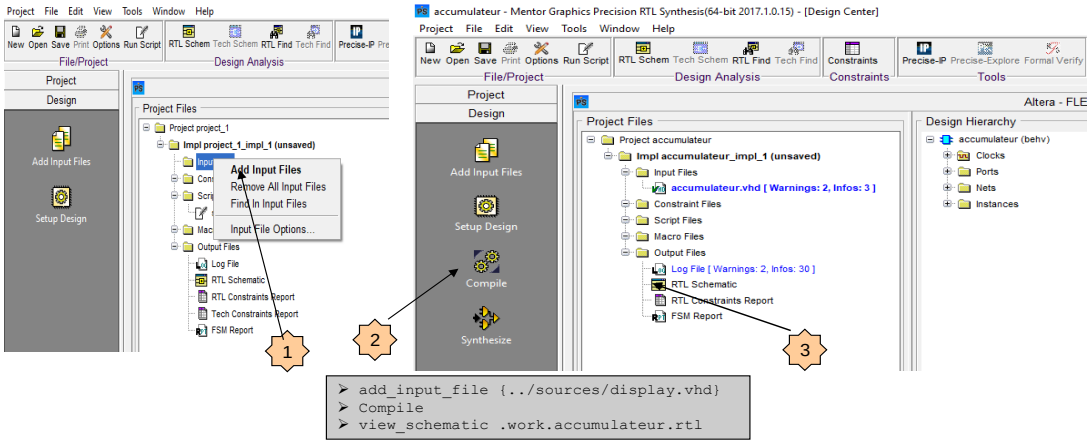


Figure 8 : sources VHDL, génération et affichage du schéma RTL.

Vérifiez d'éventuels messages d'erreurs émis par le compilateur à l'écran dans la fenêtre *transcript.*

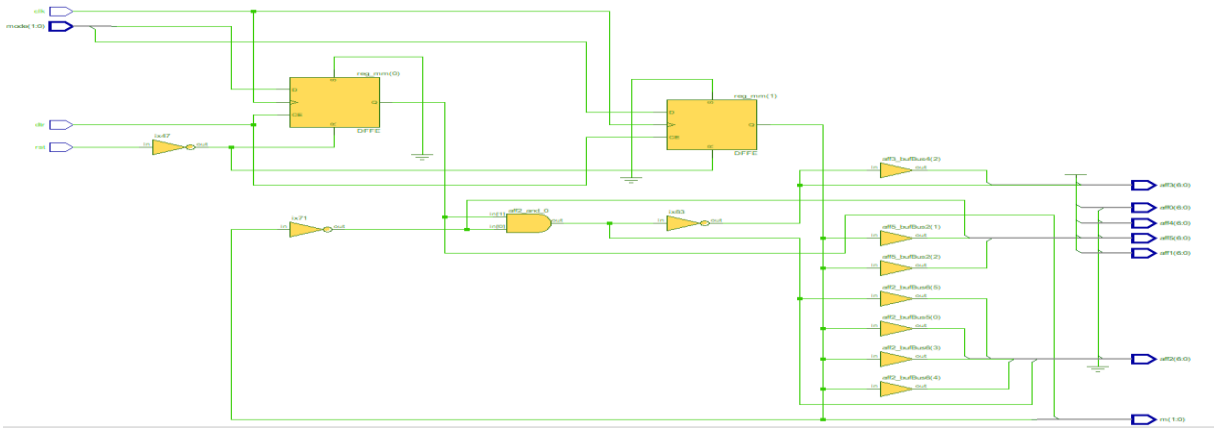


Figure 9 : Schéma RTL complet du bloc display

En cliquant sur le bouton droit de la souris sur un objet du schéma, item « *trace to hdl source* », l'outil peut établir la correspondance entre l'objet et l'expression syntaxique (le process au minimum) ayant menée à son inférence. Cette correspondance peut toutefois rester générale.

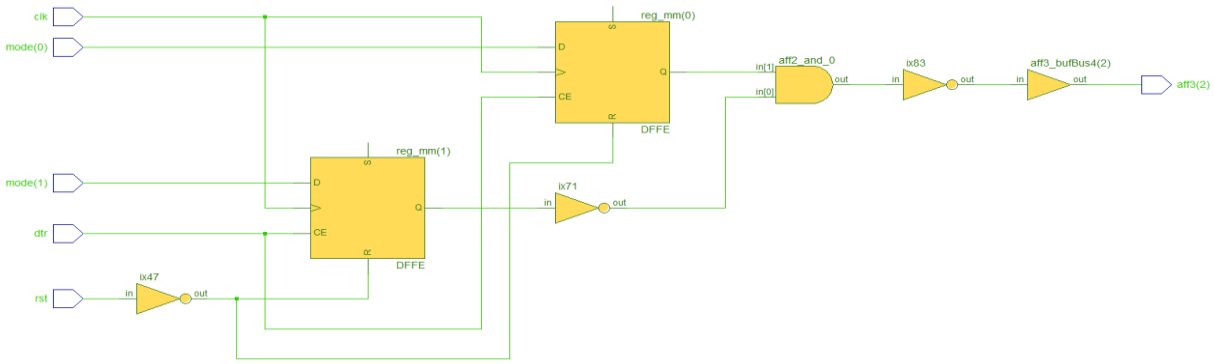


Figure 10 : Schéma RTL du bit 2 de l'afficheur 3

Il est possible de ne visualiser qu'une partie du schéma RTL. Pour observer, par exemple, les opérateurs menant à la sortie *aff3(2)* depuis les entrées du circuit. Placez-vous sur la sortie *aff3(2)* en sortie du buffer et cliquez sur le bouton droit de la souris en demandant *trace backward to input.* Ce bit peut également facilement se repérer dans la fenêtre *design hierarchy* du bloc (à gauche) en cliquant sur *ports→outputs*. Faites ensuite un *zoom fit* pour ajuster la trace à la fenêtre.

En examinant attentivement le schéma RTL, et en vous aidant des règles d'inférences exposées en cours et TD, répondez aux questions ci-dessous :

1. Identifiez les blocs combinatoires et séquentiels obtenus(inférés) sur le schéma, relativement au code comportemental.
2. Justifiez les opérateurs booléens obtenus pour le bit 2 de l'afficheur 3, en procédnt à partir de la table de vérité de ce bit.
3. Pourquoi l'instruction *if* a-t-elle inférée un registre avec enable pour le registre *mm ?*
4. Pourquoi l'instruction *case* a-t-elle inférée des opérateurs logiques et non pas de multiplexeurs pour les afficheurs ?
5. Que se passerait t il si on ajoutait l'entrée *mode* dans le process séquentiel et qu'on omettait l'attribut 'event ?

Figure 11 : Questionnaire sur la synthèse RTL

D. Optimisation logique

Poursuivez la synthèse de la description comportementale en demandant l'optimisation du circuit. Le fonctionnement de l'outil d'optimisation repose essentiellement sur les contraintes de design du bloc.

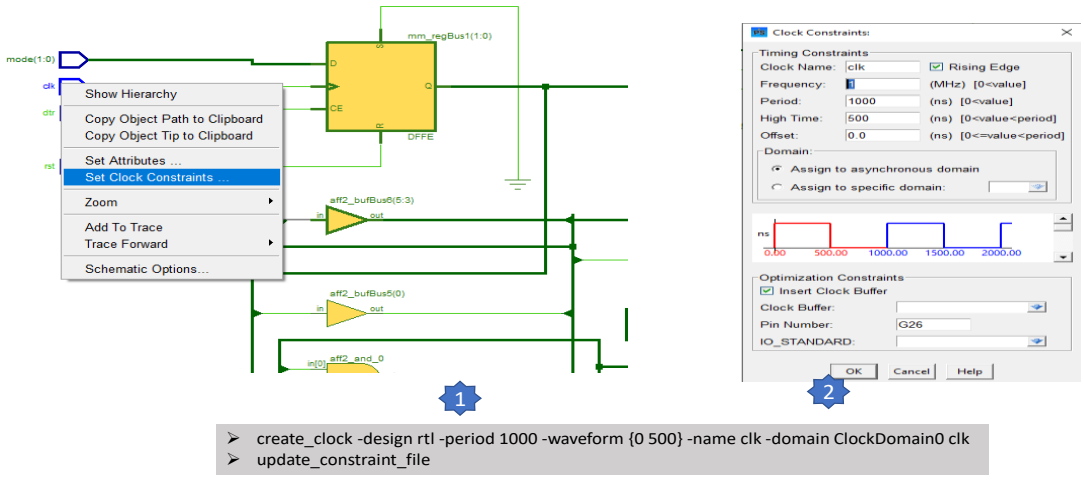


Figure 12: spécification d'une contrainte d'horloge

Fixez une contrainte d'horloge de 1 MHz en cliquant sur la patte d'horloge et en sélectionnant *set clock constraints* et en modifiant la case *frequency.*

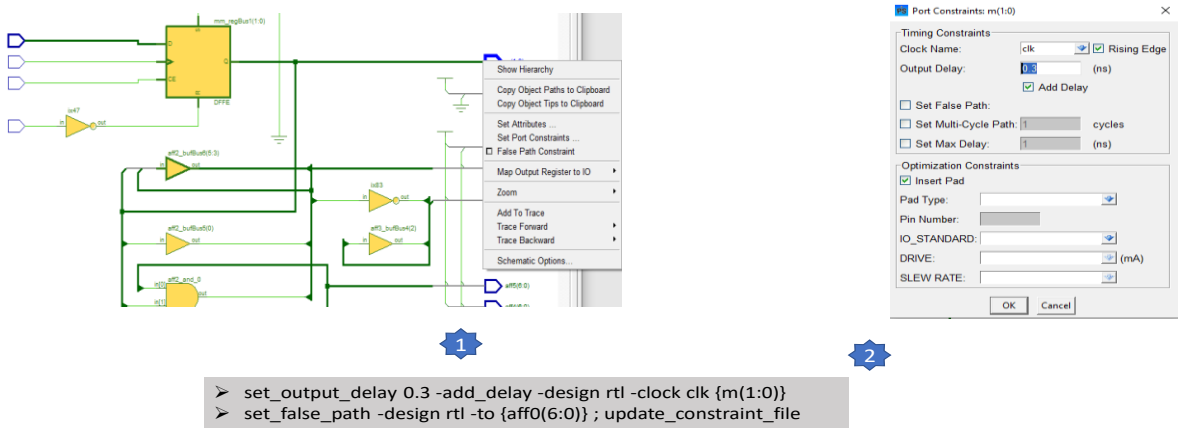


Figure 13 : Ajout d'une contrainte d'output delay sur le signal m

Spécifiez un output delay de 0.3 sur la sortie *m* (signal à destination du futur bloc *ctrlrs232c* qui sera synchrone avec le bloc *display).*

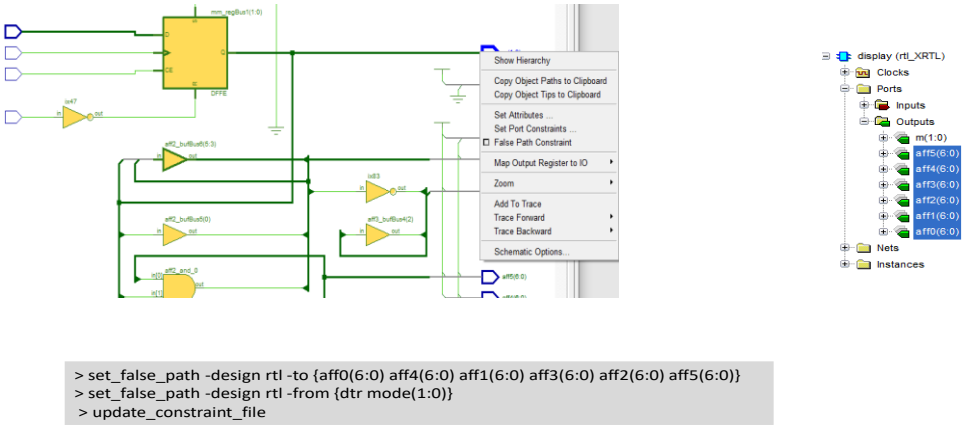


Figure 14 : Spécification d'un false path

Spécifiez finalement toutes les autres entrées et sorties comme asynchrones (en cochant set F*alse_Path* ou directement depuis le menu déroulant*)*, ces signaux étant considérés comme générés de manière asynchrone par l'opérateur.

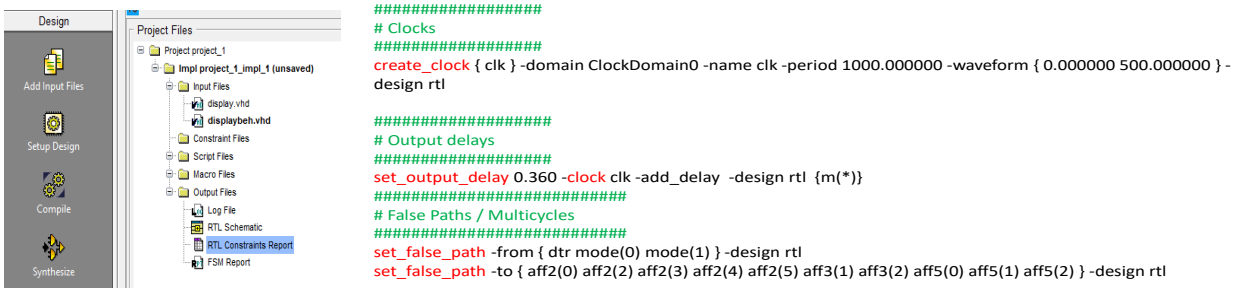


Figure 15 : Aperçu du fichier des contraintes d'optimisation

Vérifiez la prise en compte des directives d'optimisation en cliquant sur *rtl constraint report* dans le projet. Vous noterez que seuls les segments d'afficheurs variables sont contraints.

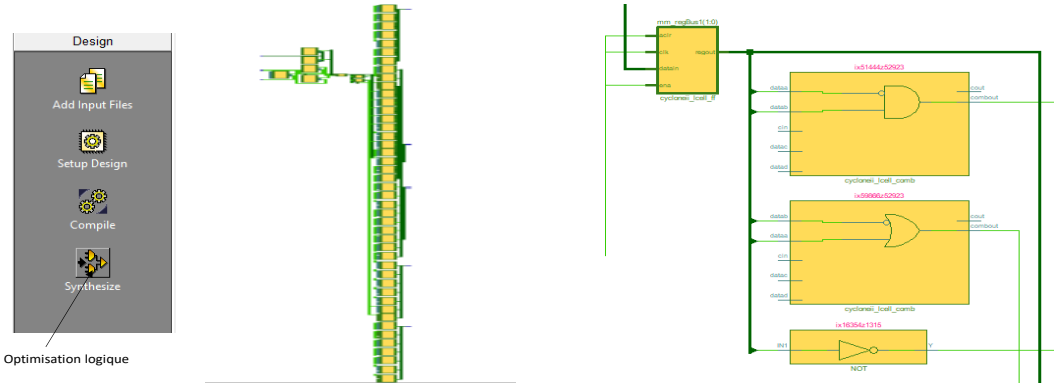


Figure 16: Schéma technologique

Cliquez sur *synthetize* et visualisez le schéma technologique (*Technology schematic* sur le panneau design center)). On notera qu'en présence de chevrons sur les signaux, cela indique que le schéma affiché n'est que partiel. Cliquez alors le bouton droit de la souris puis sur *single-page schematic* dans le popup menu qui apparait. Si vous cliquez sur *show lut information*, vous pouvez visualiser la couverture réalisée sur le DAG.

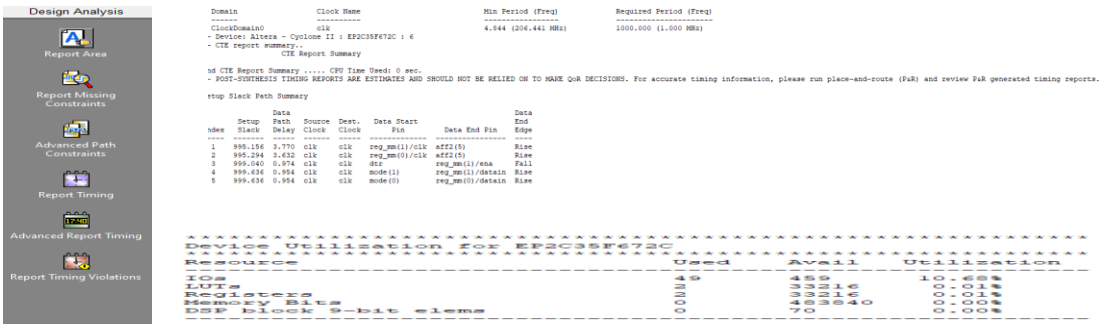


Figure 17 : Rapports de surface et de timing de précision synthesis

Cliquez sur *report area* et sur *report timing* depis le panneau *design analysis* pour obtenir les performances de l'implémentation (nombre de LE, etc) utilisés sur le circuit.

Remarque : Comme indiqué par l'outil, les résultats de timing peuvent s'avérer relativement peu fiable car certains délais, comme les temps de propagation entre cellules, ne sont qu'estimatifs.

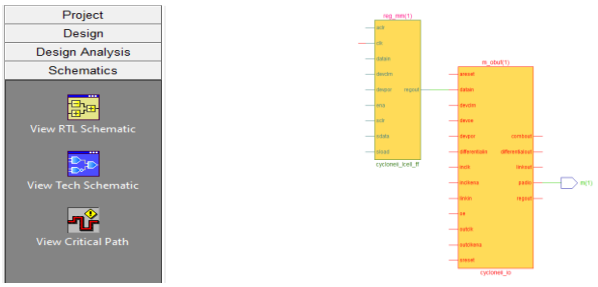


Figure 18: Chemin critique

Il est également possible de visualiser le chemin critique à partir du panneau *schematics*. A partir du schéma technologique, répondez aux questions ci-dessous :

1. Déterminez à nouveau l'emplacement des blocs séquentiels et combinatoires sur ce schéma technologique.

2. Comme sur le schéma RTL, tracez le chemin menant des entrées au bit2 de l'afficheur 3 sur ce schéma.

Figure 19: Questionnaire sur l'optimisation logique

E. Placement routage (Precision synthesis)

Le test physique du circuit sur le FPGA de la carte d'expérimentation Altera va nécessiter l'affectation des pattes du composant afin de positionner les signaux de l'interface du modèle VHDL en les reliant aux pattes du circuit (mapping) Le choix de ces affectations est guidé par l'environnement, et peut être effectué à l'aide d'attributs insérés dans le code lorsque la fréquence de modification des valeurs est rare. Dans le cas contraire, l'utilisation de directives permet de ne pas recompiler le code. Ces attributs sont définis dans la package *exemplar* mais il est néanmoins préférable de les redéfinir pour éviter des problèmes lors des simulations sous modelsim. Dans la zone de déclaration de l'architecture, ajouter les lignes :

```
attribute pin_number: string;
type defarpinnumber is array(natural range <>,natural range <>) of character;
attribute array_pin_number: defarpinnumber;
```

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
aff0[6]	Output	PIN_V13	8	B8_N0	PIN_V13	3.3-V LV...default)		24mA (default)	
aff0[5]	Output	PIN_V14	8	B8_N0	PIN_V14	3.3-V LV...default)		24mA (default)	
aff0[4]	Output	PIN_AE12	8	B8_N0	PIN_AE12	3.3-V LV...default)		24mA (default)	
aff0[3]	Output	PIN_AD11	8	B8_N0	PIN_AD11	3.3-V LV...default)		24mA (default)	
aff0[2]	Output	PIN_AC12	8	B8_N0	PIN_AC12	3.3-V LV...default)		24mA (default)	
aff0[1]	Output	PIN_AB12	8	B8_N0	PIN_AB12	3.3-V LV...default)		24mA (default)	
aff0[0]	Output	PIN_AF10	8	B8_N0	PIN_AF10	3.3-V LV...default)		24mA (default)	
aff1[6]	Output	PIN_AB24	6	B6_N1	PIN_AB24	3.3-V LV...default)		24mA (default)	
aff1[5]	Output	PIN_AA23	6	B6_N1	PIN_AA23	3.3-V LV...default)		24mA (default)	
aff1[4]	Output	PIN_AA24	6	B6_N1	PIN_AA24	3.3-V LV...default)		24mA (default)	
aff1[3]	Output	PIN_Y22	6	B6_N1	PIN_Y22	3.3-V LV...default)		24mA (default)	
aff1[2]	Output	PIN_W21	6	B6_N1	PIN_W21	3.3-V LV...default)		24mA (default)	
aff1[1]	Output	PIN_V21	6	B6_N1	PIN_V21	3.3-V LV...default)		24mA (default)	
aff1[0]	Output	PIN_V20	6	B6_N1	PIN_V20	3.3-V LV...default)		24mA (default)	
aff2[6]	Output	PIN_Y24	6	B6_N1	PIN_Y24	3.3-V LV...default)		24mA (default)	
aff2[5]	Output	PIN_AB25	6	B6_N1	PIN_AB25	3.3-V LV...default)		24mA (default)	
aff2[4]	Output	PIN_AB26	6	B6_N1	PIN_AB26	3.3-V LV...default)		24mA (default)	
aff2[3]	Output	PIN_AC26	6	B6_N1	PIN_AC26	3.3-V LV...default)		24mA (default)	
aff2[2]	Output	PIN_AC25	6	B6_N1	PIN_AC25	3.3-V LV...default)		24mA (default)	
aff2[1]	Output	PIN_V22	6	B6_N1	PIN_V22	3.3-V LV...default)		24mA (default)	
aff2[0]	Output	PIN_AB23	6	B6_N1	PIN_AB23	3.3-V LV...default)		24mA (default)	
aff3[6]	Output	PIN_W24	6	B6_N1	PIN_W24	3.3-V LV...default)		24mA (default)	
aff3[5]	Output	PIN_U22	6	B6_N1	PIN_U22	3.3-V LV...default)		24mA (default)	
aff3[4]	Output	PIN_Y25	6	B6_N1	PIN_Y25	3.3-V LV...default)		24mA (default)	
aff3[3]	Output	PIN_Y26	6	B6_N1	PIN_Y26	3.3-V LV...default)		24mA (default)	
aff3[2]	Output	PIN_AA26	6	B6_N1	PIN_AA26	3.3-V LV...default)		24mA (default)	
aff3[1]	Output	PIN_AA25	6	B6_N1	PIN_AA25	3.3-V LV...default)		24mA (default)	
aff3[0]	Output	PIN_Y23	6	B6_N1	PIN_Y23	3.3-V LV...default)		24mA (default)	
aff4[6]	Output	PIN_T3	1	B1_N0	PIN_T3	3.3-V LV...default)		24mA (default)	
aff4[5]	Output	PIN_R6	1	B1_N0	PIN_R6	3.3-V LV...default)		24mA (default)	
aff4[4]	Output	PIN_R7	1	B1_N0	PIN_R7	3.3-V LV...default)		24mA (default)	
aff4[3]	Output	PIN_T4	1	B1_N0	PIN_T4	3.3-V LV...default)		24mA (default)	
aff4[2]	Output	PIN_U2	1	B1_N0	PIN_U2	3.3-V LV...default)		24mA (default)	
aff4[1]	Output	PIN_U1	1	B1_N0	PIN_U1	3.3-V LV...default)		24mA (default)	
aff4[0]	Output	PIN_U9	1	B1_N0	PIN_U9	3.3-V LV...default)		24mA (default)	
aff5[6]	Output	PIN_R3	1	B1_N0	PIN_R3	3.3-V LV...default)		24mA (default)	
aff5[5]	Output	PIN_R4	1	B1_N0	PIN_R4	3.3-V LV...default)		24mA (default)	
aff5[4]	Output	PIN_R5	1	B1_N0	PIN_R5	3.3-V LV...default)		24mA (default)	
aff5[3]	Output	PIN_T9	1	B1_N0	PIN_T9	3.3-V LV...default)		24mA (default)	
aff5[2]	Output	PIN_P7	1	B1_N0	PIN_P7	3.3-V LV...default)		24mA (default)	
aff5[1]	Output	PIN_P6	1	B1_N0	PIN_P6	3.3-V LV...default)		24mA (default)	
aff5[0]	Output	PIN_T2	1	B1_N0	PIN_T2	3.3-V LV...default)		24mA (default)	
clk	Input	PIN_G26	5	B5_N0	PIN_G26	3.3-V LV...default)		24mA (default)	
dtr	Input	PIN_N25	5	B5_N1	PIN_N25	3.3-V LV...default)		24mA (default)	
m[1]	Output				PIN_L23	3.3-V LV...default)		24mA (default)	
m[0]	Output				PIN_L20	3.3-V LV...default)		24mA (default)	
mode[1]	Input	PIN_V2	1	B1_N0	PIN_V2	3.3-V LV...default)		24mA (default)	
mode[0]	Input	PIN_V1	1	B1_N0	PIN_V1	3.3-V LV...default)		24mA (default)	
rst	Input	PIN_W26	6	B6_N1	PIN_W26	3.3-V LV...default)		24mA (default)	

Figure 20 : Table d'affectation des pattes du bloc

Procédez ensuite à l'affectation des signaux de l'interface du bloc à une patte du FPGA en vous conformant à la figure. A titre d'exemple :

```
attribute pin_number of dtr:signal is "N25"; --SW0
attribute array_pin_number of aff0:signal is (" V13"," V14","AE12","AD11","AC12","AB12",
"AF10");
```

ce qui permet de relier le signal *dtr* au switch0 de la carte portant la référence N25, de relier le signal *aff0(6)* au segment 6 de l'afficheur HEX0 portant la référence V13, …. La sortie *mm* ne sera pas affectée. *Attention:* Toutes les chaines doivent avoir impérativement le même nombre de caractères(4 pour aff0 ici) pour un signal donné. Un espace peut être ajouté en conséquence.

Compilez et optimisez à nouveau votre design. Si vous ouvrez à nouveau le fichier de contraintes, vous observerez l'ajout des attributs de contrainte de mapping des pattes.

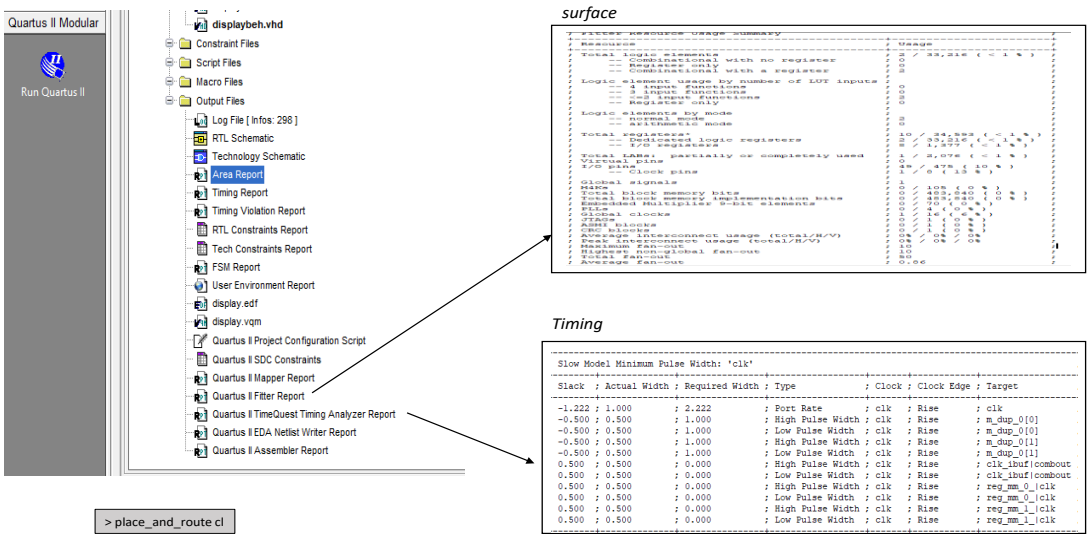


Figure 21 : Lancement de l'outil de placement routage et rapports de surface/timing

Cliquez ensuite sur *run quartus* depuis le panneau *quartus II modular* pour obtenir la synthèse physique. Cette étape se conclue par l'assemblage conduisant à lé génération du bitstream qui servira à la programmation du composant. Le rapport de surface émis par *quartus* peut être consulté depuis *précision* pour vérifier la surface et les ressources utilisées. Le rapport de timing peuvent également être consultés en cliquant sur Quartus II T*imeQuest Timing Analyser Report*, Seuls ces rapports permettent de valider l'implémentation de façon rigoureuse comme souligné précédemment.

3ème partie : Synthèse à partir d'un script de commandes

Comme pour la simulation, il est possible (et fortement recommandé) d'effectuer la synthèse à partir d'un script décrivant l'enchainement séquentiel des commandes des différents outils du flot de synthèse.

```
#script de synthèse syntheseTP1.tcl
cd C:/Users/user/Documents/Enseignements/Elec4/Asics/Tp/1_TPreglesinf/validation/precision

set_input_dir C:/Users/user/Documents/Enseignements/Elec4/Asics/Tp/1_TPreglesinf/validation/precision

setup_design -manufacturer Altera -family "CYCLONE II" -part EP2C35F672C -speed 6
setup_design -edif=true

setup_place_and_route -flow "Quartus II Modular" -command "Integrated Place and Route" -ba_format VHDL -install_dir C:/altera/13.0sp1/quartus

add_input_file {../sources/display.vhd ../sources/displaybeh.vhd}

compile
create_clock {clk} _domain clockDomain0 -name clk -period 1000.0

set_output_delay 0.36 -clock clk {m(*)}
set_false_path -from {mode(0) mode(1) dtr}
set_false_path -to {aff2(0) aff2(2) aff2(3) aff2(4) aff2(5) aff3(1) aff3(2) aff5(0) aff5(1) aff5(2)}
update_constraint_file
synthesize

place_and_route cl
```

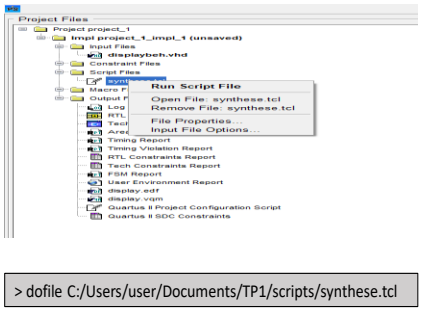


Figure 22: Inclusion et lancement d'un script de commandes

Définissez un script de synthèse *syntheseTP1.tcl* dans un fichier texte en reprenant les différentes étapes du flot de synthèse du bloc *display* à partir des commandes textuelles. On notera que la création d'un projet est automatique si celui-ci n'existe pas au lancement d'un script Evitez les copier-coller depuis le sujet. Complétez ce script pour l'affectation des pattes, en se fondant sur la forme générale de la directive suivante :

```
set_attribute  -name pin_number -value <chaine> -port <signal>
```

Par exemple, l'affectation de la patte G26 au signal *clk* s'écrira :

```
set_attribute  -name pin_number -value G26 -port clk
```

Ces contraintes peuvent être récupérées depuis le fichier de contraintes. On notera que les signaux d'un tableau doivent être affectés un par un et que le langage *tcl* est casse dépendant, contrairement au VHDL Incluez ce fichier dans le répertoire *script files* du projet sous précision synthesis, avant de lancez ce script en cliquant sur le nom du script à l'aide du bouton droit de la souris (*run tcl script)*. Vérifiez que la synthèse fournit les mêmes résultats qu'avec les menus intéractifs. Vous pouvez ensuite

4ème partie : Simulation post-routage

La simulation post routage a pour objectif de vérifier la solution issue de la synthèse, une fois placée et routée, comme celles réalisées en Elec3. Pour cela, la netlist issue de la synthèse au format VHDL est annotée avec les délais de chaque cellule et fils de routage. Cette description est générée dans le fichier *.vho* située dans le répertoire *simulation/modelsim* de l'implémentation. *(*display_impl_1). L'avantage d'effectuer cette simulation avec *modelsim* (plutôt qu'avec *quartus*) est de pouvoir réutiliser le testbench de la simulation RTL. Mais, cela demande la compilation au préalable de la bibliothèque des cellules de la technologique utilisée. Créez un nouveau fichier scriot *simupostroutage.tcl* et insérez les commandes ci-dessus, en modifiant le chemin menant au répertoire d'emplacement des fichiers sources pour le post routage (en rouge pour la commande *cd*).

```
cd C:/Users/user/< A adapter à votre projet >/precision/<nom impl>/simulation/modelsim
vlib work
vlib cycloneii
vmap cycloneii cycloneii

#compilation des bibliothèques de cellules du cyclone II
vcom -work cycloneii c:/altera/90/quartus/eda/sim_lib/cycloneii_atoms.vhd
vcom -work cycloneii c:/altera/90/quartus/eda/sim_lib/cycloneii_components.vhd

#compilation du .vho
vcom display.vho

#compilation du testbench depuis le repertoire sources
vcom ../../../../sources/testdisplay.vhd

add wave /*
#lancement du simulateur
vsim -t 1ns work.test(bench)
```

Figure 23: Script pour la simulation post routage

Sous *modelsim,* lancez votre script et observez l'apparition des signaux issues de la génération de la netlist technologique sur le waveform. Vous pouvez vérifier la conformité des signaux en sortie de la simulation post routage avec ceux issus de la simulation RTL de la 1ère partie.

Remarque : Si votre fichier de test contient une directive de configuration *for uut…* , mettez la en commentaire ou changez le nom de l'architecture en *structure.*

5ème partie : Test du circuit FPGA

Lancez l'interface de quartus II en cliquant sur l'icône bleue du projet généré par precision (.qpf) et présent dans le répertoire d'implémentation (*impl)* de votre projet. Vérifiez que quartus 13.0 est bien lancé. Dans le menu, cliquez sur *tools→chip planner.*

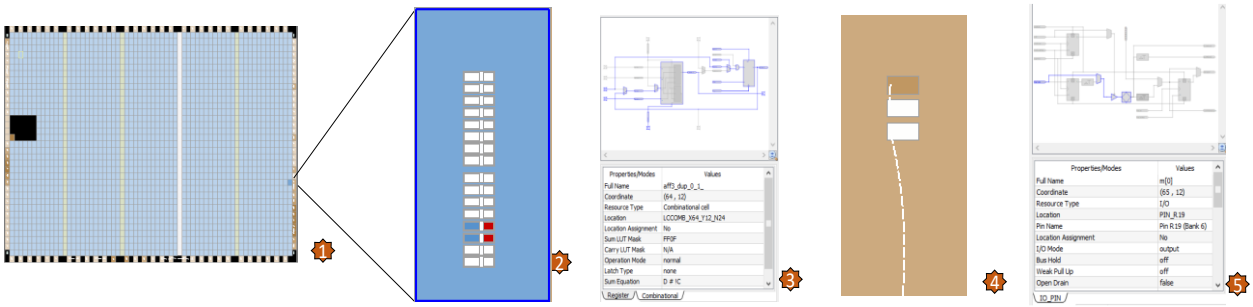


Figure 24: Floorplan du circuit

1. Une vue interne des cellules du composant permet de situer l'implémentation du bloc display (LAB en bleu foncé sur la droite).
2. En zoomant, vous pouvez observer l'emplacement des 2 cellules physiques (LE) du LAB matérialisant le bloc display sur le circuit. Ces 2 cellules physiques implémentent les 4 cellules logiques du schéma technologique (registre en bleu et combinatoire en rouge).
3. En cliquant sur une des cellules physiques, vous pouvez observer la configuration de cette cellule (Cette cellule s'affiche en grand. Si vous double cliquez). Vous pouvez alors visualiser l'équation de la partie combinatoire (*sum equation)* définissant la valeur d'un segment d'affichage tel que aff3(2).
4. Vous pouvez également cliquer sur une cellule d'E/S (en marron)
5. Vous pouvez alors visualiser le signal affecté à cette cellule (m(0) ici)

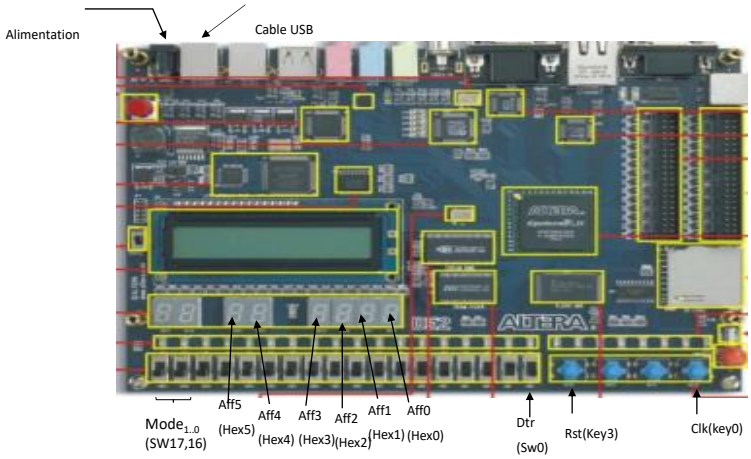


Figure 25*: Plaque d'expérimentation pour le test du circuit*

Branchez votre carte (cf figure 6). Cliquez sur *tools→programmer*. Si ce n'est pas déjà fait, cliquez sur *hardware setup* et sélectionnez *byteblaster*, Cliquez sur le bouton **start.** En vous reportant à la figure 7 validez ensuite votre circuit sur la plaque d'expérimentation en testant différentes combinaisons d'entrée de *mode* et *dtr*, combinaisons que l'on validera avec le bouton poussoir faisant office d'horloge.