Activité 4 Mini projet DMX-FPGA : Implanter un processeur dans le FPGA.Durée estimée 4hProgrammer le processeur

0 - Présentation de l'activité 4 - Console DMX :



Traitement numérique du signal. Faire évoluer le projet afin de mettre en œuvre l'affichage multiplexé. Relever des signaux commandant le multiplexage.

| Pré-requis : | \Rightarrow Avoir réaliser l'intégralité de l'activité 3 |
|-----------------|---|
| Durée estimée : | \Rightarrow 4 heures |
| Objectif : | \Rightarrow Implanter et programmer un processeu r dans un FPGA . |

Vocabulaire spécifique à l'activité :



8 Compiler, synthétiser, construire, Programmer le FPGA.

9 Mettre en œuvre les instruments de mesure virtuels et réels.

* Rappel : sous ALTIUM la feuille de schéma *.SchDoc est en haut du projet, c'est le « TOP LEVEL ».

Schéma OpenBus à dessiner:



| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 2/31 |
|------------------|--|------------|------|
|------------------|--|------------|------|



Schéma Top à modifier au cours du TP73 Console DMX:

| | Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 3 / 31 |
|--|------------------|--|------------|--------|
|--|------------------|--|------------|--------|

<u>1 Ouvrir un projet FPGA existant.</u>



1.1 Repartir d'un projet existant :

Ouvir un projet existant en utilisant la commande : File >> Open >> Projet >> FPGA Projet.

Dans l'onglet qui s'ouvre, aller dans le répertoire de travail « **TP72_ConsoleDMX** » puis choisir le projet nommé **« FPGA_ConsoleDMX.PrjFpg »**.

2 Editer et modifier le fichier OpenBus

2.1 Ouvrir un fichier OpenBus existant

Double Clic bouton gauche sur le nom du fichier OpenBus (*ConsoleDMX.OpenBus*) dans l'onglet Projets.



2.2 Placer les éléments OpenBus vsuivants afin de modifier la feuille de schéma OpenBus

$1^{\text{ière}}$ méthode :

Clic bouton droit sur la feuille de travail OpenBus et choisir dans l'onglet qui apparaît la commande **WorkspacePanels** >> **OpenBus** >> **OpenBus** Palette

| Fi <u>n</u> d Similar <u>F</u> ilter <u>P</u> lace | Objects | | | |
|--|--|---|--|---|
| <u>G</u> rids <u>V</u> iew | | NB LEDA | CTRL CONSIGLE | XSRAM |
| Workspace P Cut Copy | Ctrl+X Ctrl+C | Design Compiler Help Instruments | 2 | 68 |
| Paste | Ctrl+V | <u>O</u> penBus ▶ | OpenBus Pale <u>t</u> te | |
| Find Text | Ctrl+F | System | OpenBus <u>F</u> ilter OpenBus List | |
| | Find Similar Filter Place Grids View Workspace F Cut Copy Paste Find Text | Find Similar Objects Eilter Place Grids View Workspace Verse Cut Ctrl+X Copy Ctrl+C Paste Ctrl+V Eind Text Ctrl+F | Find Similar Objects Filter Place Grids Yiew Workspace Panels Cut Ctrl+X Copy Ctrl+C Paste Ctrl+F Eind Text Ctrl+F | Find Similar Objects Filter Place Grids Yiew Design Compiler Cut Ctrl+X Gopy Ctrl+C Paste Ctrl+V Eind Text Ctrl+F |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 4/31 | 1 |
|------------------|--|------------|------|---|
|------------------|--|------------|------|---|

$2^{ième}$ méthode :

Clic bouton gauche en bas de la feuille de travail du projet, puis choisir l'icône Openbus Palette



Le tableau ci-dessous identifie les composants Openbus à placer sur le bus périphérique

| WB_PRTIO_I | Port IO | Permet de commander les digits unité, dizaine et centaine de l'afficheur à anode commune. |
|------------|-----------------|---|
| WB_PRTIO_2 | Port IO | Permet de commander les segments (a, b, c, d, e, f et g) de l'afficheur à anode commune. |
| | Dual Timer Unit | Timer qui permet générer une interruption pour piloter l'affichage multiplexé. |

=> Placer les composants sur la feuille de schéma OpenBus



| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 5/31 |
|------------------|--|------------|------|
|------------------|--|------------|------|

=> Ajouter un port OpenBus au composant WB_INTERCON_2 :





=> Dessiner les liens entre les éléments OpenBus :



| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 6/31 |
|------------------|--|------------|------|
|------------------|--|------------|------|



WB_PRTIO_1

0

=> Paramétrer et renommer le composant WB_PRTIO_1

Clic bouton droit sur le composant WB_PRTIO_1, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Port IO**

| Configure OpenBus Port IO | ? × | | Paramétrer le port A en sortie sur 3 bits Configure OpenBus Port IO |
|---------------------------------|----------------|---|--|
| Signals | | | Signals |
| Name Kind | Add | | Name Kind PA(2,0) Out |
| PA[70] In/Out | Move Down | | - m |
| | D | | |
| | Remove | г | |
| | Move Up | | Renommer en PRTIO_DIG |
| | Paste | | |
| General Properties | | L | |
| Component Designator WB PBTID 1 | | Г | |
| | | | Cliquer sur OK pour valider les |
| Interface Type Signal Harnesses | - | | changements |
| | | - | |
| Manage Signals | <u>C</u> ancel | | |
| | | | |

=> Paramétrer et renommer le composant WB_PRTIO_2

Clic bouton droit sur le composant WB_PRTIO_2, puis dans l'onglet qui apparaît sélectionner **Configure OpenBus Port IO**

| Configure OpenBus Port | Ю | ? × | Paramétrer le port A en sortie sur 7 bits Configure OpenBus Port IO | |
|------------------------|------------------|---------------------|--|----|
| Signals | | | Signals | |
| Name | Kind | Add | Name Kind PA[60] Out | |
| PA[70] | In/Out | Move Down Remove | | |
| | | Move Up Paste | Renommer en PRTIO_SEG | |
| General Properties | | | | _ |
| Component Designator | WB_PRTIO_2 | Visible | Cliquer sur OK pour valider le | es |
| Interface Type | Signal Harnesses | • | changements | |
| Manage Signals | <u>_</u> K | Cancel | | |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 7/31 |
|------------------|--|------------|------|

=> Paramétrer le processeur comme ci-dessous :



Clic bouton droit sur le composant port Processeur \square , puis dans l'onglet qui apparaît sélectionner **Configure TSK3000A_1**



Cliquer sur OK pour valider les changements

Clic bouton gauche sur le Manage Signal, puis dans l'onglet qui apparaît sélectionner Interrupts

=> Visualiser le plan des interruptions du processeur comme ci-dessous :

| | TSK3000A(TSK3000A 1) | |
|---------------------------------------|-----------------------------|--------|
| lame | Kind and Polarity Interrupt | |
| Serial Communications Port(CTRL_UART) | | E |
| | | |
| INT_00 | Rising edge INT_10 | |
| INT_01 | Rising edge INT_11 | |
| Dual Timer Unit(TMR3_W_1) | | |
| INT_O | Rising edge Not Connected | > |
| 🛛 🚾 Interrupt pins | | |
| | Not Exported | |
| | Not Exported | - |
| | ОК | Cancel |
| | | |
| Cliquer sur Not Connected pour cl | hoisir INT 12 | |
| Dual Times Unit(TMP2 W/ 1) | | |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 8 / 31 | |
|------------------|--|------------|--------|--|
|------------------|--|------------|--------|--|

=> Visualiser le plan mémoire du processeur comme ci-dessous :

Clic bouton droit sur le port Processeur TSK3000, puis dans l'onglet qui apparaît sélectionner **Configure Processor Memory**



Cliquer sur OK pour valider les changements

=> Visualiser le plan mémoire des périphériques comme ci-dessous :

Clic bouton droit sur le port Processeur TSK3000, puis dans l'onglet qui apparaît sélectionner **Configure Processor Peripherals**

| Configure Peripherals | | | [? <mark>×</mark> | | |
|---|-------------|-------------------------------|----------------------------|--|--|
| Memory Architecture Defined Peripheral Devices | | | | | |
| 0xffff_fff | _ | OxFFFF_FFFF | | | |
| Processor I/O Space | OxFFFF_FFFF | TMR3_W_1 WB_INTERCON_2 | 0xFF06_0007 0xFF06_0000 | | |
| IO Port of the processor | 0xFE00-0000 | PRTIO_DIG WB_INTERCON_2 | 0xFF05_0000 0xFF05_0000 | | |
| | OXFEEF_FFFF | CTRL_CONSOLE WB_INTERCON_2 | 0xFF04_0003 0xFF04_0000 | | |
| | | NB_LEDS WB_INTERCON_2 | 0xFF03_001F 0xFF03_0000 | | |
| External-Memory Space | | CTRL_UART WB_INTERCON_2 | 0xFF02_000F 0xFF02_0000 | | |
| | | CTRL_SPI WB_INTERCON_2 | 0xFF01_0007 0xFF01_0000 | | |
| | 0x00FF_FFFF | PRTIO_SEG WB_INTERCON_2 | 0xFF00_0000 0xFF00_0000 | | |
| Internal-Memory Where the boot code resides | | | 0xFEFF_FFFF 0x0100_0000 | | |
| | 0x0000 0000 | | 0x0000 0000 | | |
| 0x0000_0000 | | 0x0000_0000 | | | |
| Name Address | ∠ Size | Туре | Interrupts | | |
| PRTIO_SEG 0xFF000000 | 0x0001 | Peripheral | | | |
| CTRL_SPI 0xFF010000 | 0x0008 | Peripheral | 50.54 | | |
| UTRL_UART UXFF020000 | 0x0010 | Peripheral | EU,EI | | |
| CTRL CONSOLE 0xFF040000 | 0x0004 | Peripheral | | | |
| PRTIO_DIG 0xFF050000 | 0x0001 | Peripheral | | | |
| TMR3_W_1 0xFF060000 | 0x0008 | Peripheral | E2 | | |
| Generate following files into the subproject(s) at FPGA Project compilation Inardware.asm (Assembly File) Inardware.h (C Header File) | | | | | |
| K | | Configure Memory | OK Cancel | | |
| | Ne | pas oublier de cocher | | | |

Cliquer sur OK pour valider les changements

2.3 Sauvegarder le fichier OpenBus

Clic bouton droit sur le nom du fichier OpenBus (*ConsoleDMX.OpenBus*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save As** avec le nom *ConsoleDMX.OpenBus* dans le répertoire de travail « **TP72_ConsoleDMX** ».

3 Editer et modifier le « TOP » schéma

3.1 Ouverture du fichier schéma en tête du projet FPGA :

=> Ouvrir un schéma existant :

Double Clic bouton gauche sur le nom du fichier schéma (*Top_ConsoleDMX.SchDoc*) dans l'onglet Projets.

| Projects | ▼ Ø × |
|--------------------------------------|-------------|
| Workspace1.DsnWrk | ▼ Workspace |
| FPGA_ConsoleDMX.PrjFpg | Project |
| ● File View ○ Structure Ed | ditor 🔕 💷 |
| 🗆 🔊 FPGA_ConsoleD 🕅 | X.PrjFpg |
| 🖃 🚞 Source Documer <mark>i</mark> ts | |
| 🖃 🖂 Top_Console[<mark>1</mark> M | X.SchDoc |
| ConsoleDMX.0 |)penBus |
| 🕀 🚞 Settings | |
| 🕀 🖿 Generated (NB3000 | AL_02) |

3.2 Placer dans le schéma le symbole créé à partir du fichier OpenBus :

Dans la barre de menu choisir la commande **Design** >> **Create Sheet Symbol From Sheet or HDL**



=> Cliquer sur le nom du fichier « *ConsoleDMX.OpenBus* »:

| Choose Document to Place | | ? × |
|------------------------------|--|-----|
| Document Name | △ Document Path | |
| 🖃 🚞 OpenBus System Documents | | |
| ConsoleDMX.OpenBus | D:\ALTIUM_RNR_Nov2012\TP72_ConsoleDMX\ | |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 11 / 31 |
|------------------|--|------------|---------|
|------------------|--|------------|---------|

=> Organiser le corps du symbole comme ci-dessous en déplaçant les ports bleu et jaune :



interruption.

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 12/31 |
|------------------|--|------------|-------|
|------------------|--|------------|-------|

Schéma TOP complet à obtenir :



NB3000 Console DMX avec visualisation des signaux SPI

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 13 / 31 |
|------------------|--|------------|---------|
|------------------|--|------------|---------|

3.3 Placer et paramétrer les composants sur le schéma :

 \Rightarrow Placer dans le schéma les éléments suivants :

| Description | Nom de la fonction | bibliothèque |
|--|--------------------|--------------------------------|
| Horloge de référence de 50 MHz | CLOCK_REFERENCE | FPGA NB3000 Port-Plugin.IntLib |
| Diviseur générique par 256 | CDIV256DC50 | FPGA Generic.IntLib |
| Afficheur Multiplexé 3 digits à 7 segments | AFF_7SEG_BUS | EXT_DE0_NANO.SchLib |
| | | |

3.3.1 Placer l'afficheur multiplexé sur le schéma



- \Rightarrow cliquez sur **Librairies** sur le bord droit de l'écran
- \Rightarrow Sélectionnez la bibliothèque du composant
- \Rightarrow Sélectionnez le composant
- \Rightarrow Placer le composant

| Libraries 🔻 🖉 | x |
|-------------------------------|---|
| Libraries Search ce AFF_7SEG_ | |
| SEXT_DE0_NANO.SchLib | |
| × | |
| Component Name | |
| 🕕 AFF_7SEG_PIN 🕕 BP_0 | |
| | |
| 22 components | - |
| SECTOR | ^ |
| | |
| Cathode constants | |

Double clic sur le composant AFF_7SEG_BUS puis clic sur l'onglet Edit Pins

| Component P | in Editor | | | | | | ? <mark>X</mark> |
|-----------------|-----------|-------|-------------------------------|----------|-----------------------|--------|------------------|
| Designator / | Name | Desc | Туре | Owner | Show | Number | Name |
| DIG[20] | DIG[20] | | Input | 1 | ✓ | | ~ |
| SEG[60] | SEG[60] | | Input | 1 | ✓ | | ✓ |
| SEL_DP | SEL_DP | | Input | 1 | ✓ | | ✓ |
| Add Remove Edit | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | Suppr | rimer l'entrée « er sur OK | SEL_DP » | puis | | |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 14 / 31 |
|------------------|--|------------|---------|
|------------------|--|------------|---------|

3.3.2 Placer le diviseur comme sur le schéma comme ci-dessous :

Le diviseur va permettre de définir la fréquence de rafraîchissement de l'affichage multiplexé.



 \Rightarrow Placer le composant diviseur sur le schéma

 \Rightarrow Relier les fils entre eux

3.4 Numérotation des composants :

Utiliser la fonction automatique :

 $\Rightarrow Menu : TOOLS$ $\Rightarrow Commande : Annotate Schematics Quietly...$

Annotate Schematics Quietly...

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 15/31 |
|------------------|--|------------|-------|
|------------------|--|------------|-------|

4 Définir les fichiers des contraintes.

Les fichiers des contraintes décrivent notamment la connexion broche à broche des fonctions implémentées dans le FPGA. Comme nous travaillons toujours avec la Nanoboard 3000AL2 il est plus rapide de reprendre toujours le même fichier de contraintes fourni par ALTIUM.

Autres rôles des fichiers des contraintes:

 \Rightarrow paramétrer les broches spécifiques tels que l'horloge.

 \Rightarrow si nous décrivons un projet à une autre carte que la Nanoboard il faudra alors créer les fichiers contraintes propres à cette carte.

4.1 Le fichier de contrainte sur les horloges :

 \Rightarrow Modifier le fichier de contrainte du projet :

Double clic sur le nom du fichier contrainte (Contraint1.Contraint)

| Projects | ▼ 🖉 × | 🔦 Constraint1.Constraint |
|-------------------------------|-------------------------------|---|
| Workspace1.DsnWrk | Workspace | |
| FPGA_ConsoleDMX.PrjFpg | Project | ;Constraints File ; Device : |
| File View O Structure Editor | •ال | ; Board : ; Project : |
| 🗆 🔊 FPGA_ConsoleDMX.PrjFpg * | | |
| 🗆 🚞 Source Documents | | ; Created 10/11/2012 |
| 🗆 🗔 Top_ConsoleDMX.SchDoc | D | · · · · · · · · · · · · · · · · · · · |
| 🔒 ConsoleDMX.OpenBus | | |
| 🖃 📖 Settings | | · · · · · · · · · · · · · · · · · · · |
| 🖃 🚞 Constraint Files | | Record=FileHeader Id=DXP Constraints v1.0 |
| 🔦 Constraint1.Constraint | B | · · · · · · · · · · · · · · · · · · · |
| 🛨 🛄 Harness Definitions Files | | |

 \Rightarrow Pour compléter ce fichier des contraintes nous allons copier les lignes suivantes dans le fichier de contrainte.

| / | | | |
|--------------------|--------------------------|----------------------------|---|
| Record=FileHeader | Id =DXP Constrain | ts v1.0 | |
| ; | | | |
| Record=Constraint | TargetKind=Port | TargetId=CLK_REF | FPGA_CLOCK=TRUE |
| Record=Constraint | TargetKind=Port | TargetId=CLK_REF | FPGA_CLOCK_FREQUENCY=20 Mhz |
| Record=Constraint | TargetKind=Port | TargetId=CLK_BRD | FPGA_CLOCK=TRUE |
| Record=Constraint | TargetKind=Port | TargetId=CLK_BRD | FPGA_CLOCK_FREQUENCY=50 Mhz |
| Record=Constraint | TargetKind=Port | TargetId=JTAG_NEXUS_TCK | FPGA_CLOCK=TRUE |
| Record=Constraint | TargetKind=Port | TargetId=JTAG_NEXUS_TCK | FPGA_CLOCK_FREQUENCY=1 Mhz |
| ; | | | · · · · · · · · · · · · · · · · · · · |
| | | | |
| ; | | | |
| ; Peripheral Board | - General I/O A | fficheur Multplexé 3 Digit | s |
| ; | | | |
| Record=Constraint | TargetKind=Port | TargetId=SEG[60] | <pre>FPGA_PINNUM= AB3,AD1,AE2,AC2,AF3,AH3,AB5</pre> |
| Record=Constraint | TargetKind=Port | TargetId=DIG[20] | FPGA_PINNUM=W10,W8,AC3 |
| Record=Constraint | TargetKind=Port | TargetId=PB0 | FPGA_PINNUM=AA13 |
| Record=Constraint | TargetKind=Port | TargetId=PB2 | FPGA_PINNUM=AF9 |
| Record=Constraint | TargetKind=Port | TargetId=PB4 | FPGA_PINNUM=AH8 |
| Record=Constraint | TargetKind=Port | TargetId=PB6 | FPGA_PINNUM=AG7 |
| ; | | | |

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 16/31 | I |
|------------------|--|------------|-------|---|
|------------------|--|------------|-------|---|

Clic bouton droit sur le nom du fichier contrainte (*Contraint1.Contraint*) dans l'onglet Projets et choisir la commande sauvegarder le document **Save** dans le répertoire de travail « **\TP72_ConsoleDMX** ».

<u> Après :</u>

| Projects 🔻 🖉 🗙 | Sconstraint1.Constraint |
|--------------------------------|--|
| Workspace1.DsnWrk | |
| FPGA_ConsoleDMX.PrjFpg Project | ; Constraints File ; Device : |
| File View O Structure Editor | ; Board : |
| | ; Project : ; |
| | ; Created 24/11/2012 |
| Top ConsoleDMX SchDoc | |
| A ConsoleDMX.OpenBus | |
| E Settings | 7 |
| E Constraint Files | Record=FileHeader Id=DXP Constraints v1.0 |
| NB 3000AL.02. Constraint | · · · · · · · · · · · · · · · · · · · |
| 🖏 Constraint1.Constraint 🛛 🖹 | |
| 🕀 🛄 Harness Definitions Files | Record=Constraint TargetKind=Port TargetId=CLK_REF FPGA_CLOCK=TRUE |
| 🗉 🚞 Generated (NB3000AL_02) | Record=Constraint TargetKind=Port TargetId=CLK_REF FPGA_CLOCK_FREQUENCY=20 Mhz |
| | Record=Constraint TargetKind=Port TargetId=CLK_BRD FPGA_CLOCK=TRUE |
| | Record=Constraint TargetKind=Port TargetId=CLK_BRD FPGA_CLOCK_FREQUENCY=50 Mhz |
| | Record=Constraint TargetKind=Port TargetId=JTAG_NEXUS_TCK FPGA_CLOCK=TRUE |
| | Record=Constraint TargetKind=Port TargetId=JTAG_NEXUS_TCK FPGA_CLOCK_FREQUENCY=1 Mhz |
| | 2 |
| | |
| | ; Peripheral Board - General I/O Afficheur Multplexé 3 Digits |
| (| Record=Constraint TargetKind=Port TargetId=SEG[60] PPGA_PINNUM= AB3, AD1, AE2, AC2, AF3, AH3, AB5 Record=Constraint TargetKind=Port TargetId=DIG[20] PPGA_PINNUM=W10, W8, AC3 |
| | |

Tableau de correspondance entre l'afficheur multiplexé câblé sur la carte de prototype PB30 et le brochage du FPGA :

| Nom netlist | | TargetId | FPGA_PINNUM | Carte PB30 |
|-------------|---------------|----------|-------------|------------|
| sur schéma | | | | |
| SEG[6] | Segment a | SEG[6] | AB5 | IO17 |
| SEG[5] | Segment b | SEG[5] | AH3 | IO19 |
| SEG[4] | Segment c | SEG[4] | AF3 | IO21 |
| SEG[3] | Segment d | SEG[3] | AC2 | IO23 |
| SEG[2] | Segment e | SEG[2] | AE2 | IO25 |
| SEG[1] | Segment f | SEG[1] | AD1 | IO27 |
| SEG[0] | Segment g | SEG[0] | AB3 | IO29 |
| | | | | |
| DIG[0] | Digit unite | DIG[0] | AC3 | IO31 |
| DIG[1] | Digit dizaine | DIG[1] | W8 | IO33 |
| DIG[2] | Digit | DIG[2] | W10 | IO35 |
| | centaine | | | |
| | | | | |
| SPI_CLK | | PB6 | AG7 | IO7 |
| SPI_DOUT | | PB4 | AH8 | IO5 |
| SPI_DIN | | PB2 | AF9 | IO3 |
| SPI_CS | | PB0 | AA13 | IO1 |

| Formation ALTIUM Act4_DMX-FPGA_Affichage multiplexe_timer 10/06/2013 1/ | Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 17/31 |
|---|------------------|--|------------|-------|
|---|------------------|--|------------|-------|

5 Ouverture du projet embarqué.

Programmation du code C : SOFTWARE DESIGN FLOW

5.1 Ouvrir un projet embarqué existant

Le projet a été ouvert lors de l'ouverture du projet « FPGA_ConsoleDMX.PrjFpg ».

6 Modification du fichier « Software Platform » Mise en place des API

API : Application **P**rogramming Interface Une **interface de programmation** est une interface fournie par un programme informatique. Elle permet l'interaction entre le programme et les couches matérielles de bas niveaux.

Un exemple : Le paramétrage d'une liaison série : débit binaire (9600 bauds), nombre de bits par octet (8), type de parité (aucune), nombre de bits de stops (1).

Pour plus d'information lire : Introduction ti the Software platform

🔻 🦉 🗙 Projects Workspace1.DsnWrk Workspace Ŧ Embedded_ConsoleDMX.PrjEmb Project - 🕗 File View Structure Editor ۲ FPGA_ConsoleDMX.PrjFpg 🖃 🛄 Source Documents 🖃 🖵 Top_ConsoleDMX.SchDoc 🗛 ConsoleDMX.OpenBus P 🕀 🚞 Settings 🕀 🚞 Generated (NB3000AL 02) Embedded_ConsoleDMX.PrjEmb/ 🖃 🛄 Header Documents hardware.h E 🔲 Source Documents P 👩 main.c Software Platform1.SwPlatform Ē 🛨 🛄 Generated

Double Clic bouton gauche sur le nom du fichier (*Software Platform1.SwPlatform*)

Pour obtenir la couche supérieure du port Digital I/O ⇒ *Clic bouton droit* sur l'icône **General Purpose I/O Port**

| vant : | | / | | | |
|-------------------|---------------------|---------------------|-----------------------|--|----------------------|
| Device Stacks | | | | | |
| Custom Instrument | General Purpose I/0 | General Purpose I/0 | LED Controller Driver | ADC0845021 ADC Driver DRV_ADC0845021_1 SPI Driver DRV_SPI_1 SPI Master Controller | TMR3 Dual Timer Unit |
| CTRL_CONSOLE | Port PRTIO_DIG | Port PRTIO_SEG | NB_LEDS | CTRL_SPI SPI | TMR3_W_1 |
| • | | m | | | |

Puis dans l'onglet qui s'ouvre,

cliquer sur Grow Stack Up puis cliquer sur l'icône

Répéter la commande sur les icônes General Purpose I/O Port et TMR3 Dual Timer Unit

GPIO Port Driver

| Α | nr | ٠Ò٢ | • | |
|---|-------|-----|---|--|
| 1 | ν | CD | ٠ | |
| | | | | |

| | | | | ADC0845021 ADC Driver DRV_ADC084S021_1 | |
|-------------------|---------------------|---------------------|-----------------------|--|----------------------|
| Custom Instrument | GPIO Port Driver | GPIO Port Driver | LED Controller Driver | SPI Driver | TMR3 Dual Timer |
| DRV_INSTRUMENT_1 | DRV_IOPORT_1 | DRV_IOPORT_2 | DRV_LED_1 | DRV_SPI_1 SPI | DRV_TMR3_1 |
| Digital I/O | General Purpose I/O | General Purpose I/O | LED Controller | SPI Master Controller | TMR3 Dual Timer Unit |
| CTRL_CONSOLE | PRTIO_DIG | PRTIO_SEG | NB_LEDS | CTRL_SPI §PI | TMR3_W_1 |
| • | | | | | • |

Ces interfaces donnent accès aux différentes fonctions prédéfinies (API) :

- commande des digits de l'afficheur multiplexé (DRV_IOPORT_1)
- commande des segments de l'afficheur multiplexé (DRV_IOPORT_2)
- gestion du timer TMR3 (DRV_TIMER3_1)



7 Ajout du fichier C principal

7.1 Créer le tableau de codage BCD vers afficheur à 7 Segments

L'afficheur multiplexé est composé par un afficheur à anodes communes composé de 4 digits (HDSP-B08G).

Schéma :



7.1.1 Commande des DIGITS

Quel niveau logique doit-on appliquer sur l'entrée DIG[x] pour sélectionner un afficheur ? NL0 qui permet de rendre le transistor passant

Compléter le tableau ci-dessous :

| | DIG[2] | DIG[1] | DIG[0] | DIG[20] |
|--------------------------|--------|--------|--------|-------------|
| Sélection digit Unité | 1 | 1 | 0 | 0x06 |
| Sélection digit Dizaine | 1 | 0 | 1 | 0x05 |
| Sélection digit Centaine | 0 | 1 | 1 | 0x03 |

Exemple commande du digit Unité :

ioport_set_value(drv_ioport_dig, 0, 0x06);

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 20/31 |
|------------------|--|------------|-------|

7.1.2 Commande des segments à anode communes

Quel niveau logique doit-on appliquer sur l'entrée d'un SEG[x] pour allumer un segment ?

Compléter le tableau ci-dessous :



| Nombre BCD | Segments | | | | | | | |
|------------|----------|---|---|---|---|---|---|-------------|
| | а | b | с | d | e | f | g | SEG[60] |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0x01 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0x4F |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0x12 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0x06 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0x4C |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0x24 |
| 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0x60 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0x0F |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x88 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0x04 |

Déclaration du tableau BCD_7Seg_Table en langage C :

// Tableau conversion BCD -> 7 Segments

unsigned char BCD_7Seg_Table[16]={0x01,0x4F,0x12,0x06,0x4C,0x24,0x60,0x0F,0x00,0x04, 0XFF,0xFF,0xFF,0xFF,0xFF,0xFF};

Exemple commande du digit Unité :

ioport_set_value(drv_ioport_seg, 0, BCD_7Seg_Table[Unite]);

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 21/31 |
|------------------|--|------------|-------|
|------------------|--|------------|-------|

7.2 Algorithme de l'interruption extérieure

Les variables utilisées dans l'algorithme suivant sont :

| Etat_Actuel : | variable représente l'état de la machine d'état (Etat_Digit_Unite, |
|---------------|--|
| | Etat_Digit_Dizaine, Etat_Digit_Centaine). |
| Val_Aff : | variable représente le nombre à afficher (0 à 255). |
| a | |
| Centaine : | mémorise la valeur du digit centaine. |
| Reste : | mémorise un calcul intermédiaire. |
| Dizaine : | mémorise la valeur du digit dizaine. |
| Unité : | mémorise la valeur du digit unité. |

Début interruption extérieure 2

| Centaine \leftarrow Val_Aff / 100 ; | // Extrait le digit Centaine à partir de Val_Aff |
|---------------------------------------|--|
| Reste \leftarrow Val_Aff % 100 ; | // Extrait les digits Dizaine et Unité à partir de Val_Aff |
| Dizaine \leftarrow Reste / 10 ; | // Extrait le digit Dizaine à partir de Reste |
| Unité ← Reset % 10 ; | // Extrait le digit Unité à partir de Reste |

Suivant (Etat_Actuel) Faire

Cas (Etat_Actuel == Etat_Digit_Unite) Sélectionner Digit Unité; Afficher Caractère Unité; Etat_Actuel ← Etat_Digit_Dizaine;

Cas (Etat_Actuel == Etat_Digit_Dizaine) Sélectionner Digit Dizaine ; Afficher Caractère Dizaine; Etat_Actuel ← Etat_Digit_Centaine;

Cas (Etat_Actuel == Etat_Digit_Centaine) Sélectionner Digit Centaine ; Afficher Caractère Centaine; Etat_Actuel ← Etat_Digit_Unite;

Sinon Etat_Actuel ← Etat_Digit_Unite ;

FinSuivant

Fin

7.3 Coder l'algorithme du programme d'interruption en langage C

A partir de l'algorithme, coder le programme d'interruption en langage C

```
/* Déclaration de la fonction d'interruption */
void __interrupt (INTNUMBER_EXT2) it_exterieure (void)
{
   Centaine=(Val_Aff/100);
   Reste= Val_Aff%100;
   Dizaine=Reste/10;
   Unite= Reste%10;
   switch ( Etat_Actuel )
   {
       case Etat_Digit_Unite:
       {
          ioport_set_value( drv_ioport_dig, 0, UNITE); // Sélection Digit Unité
          ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Unite]);
          Etat_Actuel=Etat_Digit_Dizaine;
       break;
       case Etat_Digit_Dizaine:
       {
          ioport_set_value( drv_ioport_dig, 0, DIZAINE);
                                                      // Sélection Digit Dizaine
          ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Dizaine]);
          Etat_Actuel=Etat_Digit_Centaine;
       break;
       case Etat_Digit_Centaine:
       {
           ioport_set_value( drv_ioport_dig, 0, CENTAINE);
                                                          // Sélection Digit Centaine
          ioport_set_value( drv_ioport_seg, 0, BCD_7Seg_Table[Centaine]);
          Etat_Actuel=Etat_Digit_Unite;
       break;
       default: Etat_Actuel=Etat_Digit_Unite;
       break;
   }
}
```

| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 23 / 31 |
|------------------|--|------------|---------|
|------------------|--|------------|---------|



24/31

7.4 Copier le fichier « main.c »

Copier le fichier ci-dessous dans le fichier main.c puis enregistrer et compiler le projet.

```
/* Déclaration des fichiers d'entête */
#include <swplatform.h>
#include <stdio.h>
*
#define UNITE 0x06
#define DIZAINE 0x05
#define CENTAINE 0x03
/******
unsigned char DMX_Table[512];
                              // Déclaration d'un tableau de 512 caractères
// Tableau conversion BCD -> 7 Segments
OXFF, OxFF, OxFF, OxFF, OXFF, OxFF};
uint8_t Val_Aff=0;
//uint8 t Etat=0;
uint8_t Centaine, Dizaine, Unite, Reste;
 /* etats */
typedef enum
  Etat Digit Unite,
  Etat_Digit_Dizaine,
  Etat_Digit_Centaine,
}
etat_type;
etat_type Etat_Actuel = Etat_Digit_Unite ;
void handler_timer_a (tmr3_t *drv_tmr3_1, void *data)
{
   Centaine=(Val_Aff/100);
   Reste= Val_Aff%100;
   Dizaine=Reste/10;
   Unite= Reste%10;
   switch ( Etat_Actuel )
   {
      case Etat Digit Unite:
      {
         ioport_set_value( drv_ioport_1, 0, UNITE); // Sélecti
ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Unite]);
                                              // Sélection Digit Unité
         Etat_Actuel=Etat_Digit_Dizaine;
      break;
      case Etat Digit Dizaine:
      {
         ioport_set_value( drv_ioport_1, 0, DIZAINE);
                                                 // Sélection Digit Dizaine
         ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Dizaine]);
         Etat_Actuel=Etat_Digit_Centaine;
      break;
      case Etat_Digit_Centaine:
      {
         ioport_set_value( drv_ioport_1, 0, CENTAINE);
                                                 // Sélection Digit Centaine
         ioport_set_value( drv_ioport_2, 0, BCD_7Seg_Table[Centaine]);
         Etat_Actuel=Etat_Digit_Unite;
      break;
      default: Etat_Actuel=Etat_Digit_Unite;
      break;
   }
}
/*******
/* Déclaration de la fonction initialisation */
int Initialize(void)
                         Act4_DMX-FPGA_Affichage multiplexé_timer
                                                                     10/06/2013
Formation ALTIUM
```

```
Activité 4 Mini projet DMX-FPGA : Implanter un processeur dans le FPGA.
Durée estimée 4h
                                            Programmer le processeur
    // Ouverture du périphérique virtuel instrument INTRUMENT
drv_instrument_1 = instrument_open(DRV_INSTRUMENT_1);
    // Ouverture du périphérique DRV_LED
    drv_led_1 = led_open(DRV_LED_1);
    led_turn_all_off(drv_led_1);
                                                             // Extinction des leds
    // Ouverture du périphérique port série UART
drv_uart8_1 = uart8_open(DRV_UART8_1);
// 250kbauds, pas de parité, 8 bits de données et 2 bits de stops entre l'envoi de 2 caractères
    uart8_set_parameters(drv_uart8_1,250000, UART8_NO_PARITY,8,2);
    // Ouverture du périphérique port convertisseur ADC084S021
    drv adc084s021 1= adc084s021 open(DRV ADC084S021 1);
     // Ouverture des ports IO
    drv_ioport_1 = ioport_open(DRV_IOPORT_1);
    drv_ioport_2 = ioport_open(DRV_IOPORT_2);
                                                 // Autorisations des interruptions
    interrupts enable( );
    // Ouverture du périphérique port Timer 3
    drv_tmr3_1 = tmr3_open(DRV_TMR3_1);
    tmr3_timer_a_set_handler(drv_tmr3_1, handler_timer_a, NULL);
// Timer 3 : Gate a non Activitée, Mode Compteur, Compteur qui permet de diviser par 128
tmr3_timer_a_8bit_autoreload_mode(drv_tmr3_1, 0, 1, 128);
                                                     // Lancement du Timer3
    tmr3_timer_a_start(drv_tmr3_1);
    return 1;
}
/*********
int main(void)
    Initialize();
                                                  // Appel de la fonction d'initialisation
    while (1)
         // Lecture ADC int adc084s021(adc_t *adc, unsigned channel)
        for (uint8_t i = 0; i < 4; i++) // Lecture des 4 consignes Rouge , Vert, Bleu et Dimmer de
l'instrument virtuel
       {
            DMX_Table[i]= (char) adc084s021_read(drv_adc084s021_1,i+1); // Stockage résultat conversion
dans un tableau
                                                                 // Temporisation de 10 us entre 2 conversions
            delay us(10);
             led_set_intensity(drv_led_1, i, DMX_Table[i]);
                                                                      // Envoi commande des leds RVB
             instrument_set_value(drv_instrument_1,i,DMX_Table[i]); // Envoi résultat conversion vers
intrument virtuel
        Val_Aff = DMX_Table[0];
        uart8_putbreak(drv_uart8_1, 22);
                                                   // Envoi du break d'une durée de 22 Tbits
        uart8_putchar(drv_uart8_1, 0);
                                                  // Envoi du caractère '0'
        for (int i = 0; i < 512; i++)</pre>
                                                               // Envoi des 4 consignes du buffer DMX sur la
liaison DMX
        {
                                                                     // Test que le buffer d'émmission est vide
            while (!uart8_transmit_buf_free(drv_uart8_1));
            uart8_putchar(drv_uart8_1, DMX_Table[i]);
                                                                     // Envoi du caractère vers l'UART
        }
    }
}
/*** FIN DU PROGRAMME PRINCIPAL ***/
```

8 Compiler, Synthétiser, construire, Programmer le FPGA.

 \Rightarrow Cliquer sur Compile, cliquer sur Synthetize, cliquer sur Build.

 \Rightarrow Si une erreur apparaît vous devez la corriger en modifiant le fichier source identifié à partir du message d'erreur :

Sources d'erreurs possibles :

- \Rightarrow le fichier OpenBus
 - ⇒ le schéma TOP
 - \Rightarrow les fichiers de contraintes

| ● 🕞 <u>C</u> ompile | ▼ ● 🕞 Synthesize | ▼ ● 🕀 Build | ▼ ● 🖾 Program EPGA 🧳 |
|-----------------------------|------------------|----------------------------|--|
| | | | |
| | | Cyclone3 EP3C40F780C8N | |
| FPGA_TP2_ConsoleDMX / NB300 | 0AL_02 | | • |
| U1 NANOBOARD_INTER | FACE | CTRL_CONSOLE DIGITAL_IO | TSK3000A_1 TSK3000A |
| | | Digital VO TOI TDO | TSK-3000A 32-bit RISC Processor TDI TDO |
| | | | Compile |
| | | | mbedded_TP2_ConsoleDMX_system_TSK3000A_1.he |
| | | | |
| | | | |
| | | | |

Après avoir franchi ces trois étapes la fenêtre de résultats s'affiche

| Results Summary | 8 × | | | | |
|--|----------------------|--|--|--|--|
| | | | | | |
| Device Resources - Usage Summary | | | | | |
| Total logic elements | 6,273 / 39,600 (16%) | | | | |
| Total registers* | 2,893 / 42,205 (7%) | | | | |
| Dedicated logic registers | 2,893 / 39,600 (7%) | | | | |
| VO registers | 0 / 2,605 (0%) | | | | |
| VO pins | 134 / 536 (25%) | | | | |
| Global clocks | 2 / 20 (10%) | | | | |
| * Register count does not include registers inside R | | | | | |
| Design Statistics - Timing Summary | | | | | |
| CLK_BRD | 50.0 MHz | | | | |
| JTAG_NEXUS_TCK | 96.86 MHz | | | | |
| | | | | | |
| Show Results Summary dialog Note: The Results Summary also appears in the Output panel | | | | | |
| Arint 🗈 C <u>o</u> py <u>R</u> eport | | | | | |

=> Vous pouvez alors programmer le FPGA : cliquer sur program FPGA !



9 Mettre en œuvre les instruments de mesures virtuels et réels

9.1 Réglage de la fréquence entrante par U3 : le générateur d'horloge

 \Rightarrow Cliquer sur NanoBoard-3000AL:

| 1777 | | NB3000AL (ID: 98556610) |
|------|--------------------------|-------------------------|
| | NanoBoard-3000AL V1.0.26 | |
| | | |

 \Rightarrow Régler la fréquence à 50 MHz.

| Instrument Rack - Nanoboard Controllers | | ▼ × |
|--|---------------|-----------------------|
| JTAG 1/0 CORE (NanoBoard-3000AL) | | • • |
| PLUGINS | | CONTROL |
| Peripheral Board A | | Board View Settings |
| | | |
| CLOCK FREQUENCY | | FLASH RAM |
| 200 (Max.) 175 150 133 125 100 90 | | FPGA Boot U52 Unknown |
| 75 60 50 40 30 25 20 15 | 30,000 | Embedded U53 Unknown |
| 6 (Min.) Other Frequency Clock Control Optic | s MHz | Controller |
| | | |

Il est important de paramétrer correctement la fréquence de l'horloge car celle-ci détermine la durée Tbit sur la liaison série RS 485.

9.2 Branchement de la NB3000 avec le projecteur à Leds

Brancher l'adaptateur entre la sortie RS485 (prise RJ45) et le câble DMX. Puis relier le câble DMX au projecteur à leds adressé sur le canal n°1.



| Formation ALTIUM | Act4_DMX-FPGA_Affichage multiplexé_timer | 10/06/2013 | 28 / 31 |
|------------------|--|------------|---------|
|------------------|--|------------|---------|

9.3 Relevés des chronogrammes sur le connecteur de la PB30



Calculer la fréquence du signal entrant sur la broche ? Puis la comparer avec la fréquence de rafraîchissement ?

Freq Int_I2 = $(CLK_REF / 256) / 128 = 610.35 \text{ Hz}$ avec $CLK_REF = 20 \text{ MHz}$. Cette fréquence est 3 fois plus grande, ce qui normal car l'interruption est appelée à chaque front montant

Comparer cette fréquence de rafraîchissement avec la persistance rétinienne ? La fréquence de rafraîchissement est très supérieure à la persistance rétinienne qui est de 50Hz.

Annexe 1:

Câble pour adapter la prise RS-485 aux interfaces a 3 fiches DMX-512.



Pin 1 : DMMX OUT GROUND Pin 2 : DMX OUT DATA -Pin 3 : DMX OUT DATA +

Annexe 2:

Schéma de l'afficheur multiplexé



<u>Annexe 3:</u>



Fichier de contrainte pour la carte de prototype PB30 :

; Peripheral Board - General I/O

Annexe 4 :

Photo de l'afficheur multiplexé câblé sur la carte de prototypage PB30

