

# TP : Système sur un FPGA La plateforme ML403

1.	Introduction	2
2.	Conception du système	2
3.	Détails du système	. 15
4.	Compilation de la plateforme matérielle	. 17
5.	Les applications standalone	. 18
6.	Compilation des applications	. 19
7.	Mise à jour du bitstream	. 20
8.	Téléchargement du code via le lien JTAG	. 20
9.	Projet Test des périphériques	. 21
10.	Débuggage	. 22
11.	Ajout d'un périphérique	. 23

Module : Conception Numérique Avancée Filière : ESI Option : SE Polytech'Orléans 2006-2007

Auteur : Christophe ALAYRAC





### 1. Introduction

Nous allons voir dans ce TD la conception d'un système sur un FPGA. Les concepts développés pourront facilement être portés sur d'autre plateforme du marché.

Le travail sera réalisé sur la plateforme ML403 de Xilinx. Cette carte embarque un FPGA Virtex4FX12. Ce composant embarque :

- Un cœur de PowerPC ppc405
- Deux contrôleurs MAC Ethernet 10, 100 et 1000MHz.
- 32 blocs DSP48 (blocs de traitement du signal)
- Des blocs RAM
- ...

La plateforme (la carte) embarque de la RAM DDR, de la mémoire Flash, une interface vers une mémoire compact Flash, et différent méthode d'initialisation du FPGA à la mise sous tension.

Nous n'aurons malheureusement pas le temps de mettre en œuvre toutes ces ressources dans le détail. Nous allons nous focaliser sur la création d'un système « from scratch » comme disent nos collègues anglophones.

Ce système sera relativement basique avec le cœur du PowerPC associé à de la mémoire interne au FPGA, de la RAM DDR externe, et l'IP opb\_uartlite de Xilinx pour implémenter une liaison série.

# 2. Conception du système

Ouvrez **Platform Studio 8.1**. L'outil vous propose un Wizard pour démarrer un nouveau projet.

🔶 Xilim	c Platform Studio 🛛 🔀			
Create	new or open existing project			
BSB	<u>     Base System Builder wizard (recommended)        </u>			
i i	O Blank ≚PS project			
R R	O Open a recent project			
Brows	se for More Projects 💌			
Browse installed EDK examples (projects) <u>here</u>				





Le *Base System Builder Wizard* part en fait d'un fichier de description de la plateforme. Se fichier contient les informations sur les composants raccordés au FPGA de sorte qu'il vous sera possible de créer un système.

Cliquez simplement le bouton OK pour passer à la suite.

Create New XPS Project using BSB Wizard	
New project Project file	
C:/temp/testuart/testuart.xmp	<u>B</u> rowse
Advanced options (optional)	
	Bro <u>w</u> se
ОК	Cancel

Saisissez un nom de projet, attention aux noms avec des caractères spécifiques (espace, accents). Il est possible de préciser un site dépositaire d'IP propre ou importées d'ailleurs, comme par exemple une IP « processeur polyphase » que vous auriez développé par ailleurs et que vous souhaiteriez pouvoir intégrer à votre système.

Nous n'avons pas de site dépôt dans notre cas.





Dans la fenêtre suivante choisissez de créer un nouveau système. Sélectionnez ensuite la plateforme ML403.

🔶 Base System	Builder - Select Board	?×			
Select a target dev	velopment board:				
Select board					
💿 I would like	to create a system for the following <u>d</u> evelopment board				
Board <u>v</u> endor:	Xilinx	~			
Board n <u>a</u> me:	Virtex 4 ML403 Evaluation Platform	~			
Board <u>r</u> evision:	1	~			
Note: Visit the v	vendor website for additional board support materials.				
Vendor's Websil	te <u>Contact Info</u>				
Download Third	Party Board Definition Files				
🔘 I would like	to create a <u>s</u> ystem for a custom board				
-Board description	n				
The ML403 board is intended to showcase and demonstrate Virtex-4 technology, especially the new features being added to the FPGA. The ML403 board utilizes Xilinx Virtex 4 XC4VFX12-FF668 device. It is a demonstration platform to showcase the enormous power and flexibility of Virtex-4 FPGAs including new and improved clock technology, DSP blocks, Smart RAM blocks, advanced I/Os, embedded MACs, embedded processors, and more.					
More Info	< <u>B</u> ack <u>N</u> ext > <u>C</u> ance	el			

Pour information, si vous utilisez une carte autre que les plateformes Xilinx il faut récupérer les fichiers de descriptions de plateforme sur le site du fabricant (fichier \*.xbd) et les copier dans le répertoire EDK\board. Vous pouvez par curiosité ouvrir le fichier C:\EDK\board\Xilinx\boards\Xilinx\_ML403\data\Xilinx\_ML403\_v2\_2\_0.xbd pour vous faire une idée du format du fichier.





Dans la fenêtre qui suit vous allez configurer le processeur de votre système.

🗢 Base System	Builder - Select P	rocessor		? 🗙	
-The board you se	elected has the followin	g FPGA device:			
<u>A</u> rchitecture:	<u>D</u> evice:	Pac <u>k</u> age:	<u>S</u> peed grade:		
virtex4	xc4vfx12	ff668	-10	~	
Select the process	or you would like to use	in this design:			
		, in the Goolgn.			
O MicroBlaze	LEDS DIPe OPD OPB Abker TwiWL Ret	OP D PLU PLK PLK PLK FRAM	PPC 405		
Processor descrip	otion				
The PowerPC 405 core is a 32-bit implementation of a RISC PowerPC embedded-environment architecture. It is integrated into the Virtex-II Pro and Virtex-4 FX device using the IP-Immersion technology and supported by CoreConnect bus infrastructure and extensive IP cores for peripherals and utilities.					
More Info		< <u>B</u> ack	<u>N</u> ext>	Cancel	

Dans notre cas nous allons utiliser le cœur de powerpc embarqué dans le FPGA. Sur un Spartan 3<sup>E</sup> vous pourriez utiliser le processeur Microblaze, qui est une IP de processeur 32 bits de Xilinx.





Attention, si vous changez les horloges, le Wizard verra que vous n'utilisez pas l'horloge par défaut et utilisera une entrée horloge utilisateur du FPGA sur laquelle aucune horloge n'est câblée. Donc votre système ne fonctionnera pas à moins d'y connecter le bon signal d'horloge.

🗢 Base System Builder - Configure PowerPC 🛛 ? 🔀				
DoworD:				
Sustem wide settings				
Reference clock         Processor clock         Bus clock frequency:           frequency:         frequency:         Bus clock frequency:				
100.00 MHz 100.00 V MHz 100.00 V MHz				
Ensure that your board is configured for the specifed frequency. <u>R</u> eset polarity:       Active LOW				
Processor configuration				
(Use BRAM)				
<u>D</u> ata:				
Instruction:				
Cache setup				
More Info				

Comme les codes que nous allons mettre en œuvre ne seront pas trop encombrant nous allons nous limiter à de la mémoire interne du FPGA. (On Chip Memory). Choisissez 16k de RAM pour l'espace de données et 16k pour l'espace programme.

Le power PC ayant déjà 16k de mémoire cache embarquée (16k data et 16k instruction) nous n'avons pas besoin de cocher la case Enable Cache.





Dans l'écran suivant du wizard sélectionnez seulement l'UART OPB\_UARTLITE configurez à 9600 bauds, 8 bits un stop et pas de parité.

🗢 Base System Builder - Configure 10 Interfaces	? 🛛
The following external memory and IO devices were found on your board: Xilinx Virtex 4 ML403 Evaluation Platform Revision 1 Please select the IO devices which you would like to use: 	
Peripheral: OPB UARTLITE	Data Sheet
Baudrate (bits per seconds): 9600 V Data bits: 8 V Parity: NONE V	
Use interrupt	
LEDs_4Bit	Data Sheet
LEDs_Positions	Data Sheet
More Info	<u>C</u> ancel

Désélectionnez les autres périphériques.



TD Système sur un FPGA



🗢 Base System Builder - Configure Additional 10 Interfaces	? 🛛
The following external memory and IO devices were found on your board: Xilinx Virtex 4 ML403 Evaluation Platform Revision 1 Please select the IO devices which you would like to use: - IO devices	
Push_Buttons_Position	Data Sheet
	Data Sheet
SysACE_CompactFlash	Data Sheet
DDR_SDRAM_64Mx32	Data Sheet
Peripheral: PLB DDR	Note
Use interrupt	
More Info	<u>C</u> ancel

Sélectionnez dans cet écran la mémoire DDR externe au FPGA. Le wizard inclura alors au design une interface entre le bus PLB (peripheral local bus) du powerpc et la mémoire DDR.

Nous pourrons donc stocker notre code en mémoire interne ou en mémoire externe.





🔶 Base System Builder - Add Interna	ll Peripherals	? 🛛
Add other peripherals that do not interact with "Add Peripheral" button to select from the list o If you do not wish to add any non-10 periphera	off-chip components. Use If available peripherals. Is, click the "Next" button	the <u>A</u> dd Peripheral
Peripherals plb_bram_if_cntlr_1 Peripheral: PLB BRAM IF CNTLR Memory <u>s</u> ize: 16 KB		<u>R</u> emove Data Sheet
More Info	< Back N	ext > Cancel

Vous pourriez ici rajouter d'autres périphériques comme un timer par exemple.





🔷 Base Sys	tem Builder - Software Setup	? 🗙
-Devices to	use as standard input and standard output	
STD <u>I</u> N:	RS232_Uart	
STD <u>O</u> UT:	RS232_Uart	
_Sample app	plication selection	
Select the include a lin	sample C application that you would like to have generated. Each application will nker script.	
Memor	ny <u>t</u> est	
Illustrati	e system aliveness and perform a basic read/write test to each memory in your sys	tem
✓ Periphe	eral selftest	
Perform	a simple self-test for each peripheral in your system.	
<u>M</u> ore Info	< <u>B</u> ack <u>N</u> ext > <u>C</u> an	cel

Sélectionnez l'UART comme interface standard d'entrée et de sortie (stdin et stdout), pour que les fonctions printf et scanf utilisent le port série.

Le wizard va également créer un projet de test permettant de tester notre système.





🔶 Base Syster	m Builder - Configure Memory Test Application	? 🗙				
The simple Memory Test application will illustrate system aliveness and perform a basic read/write test to your memory devices.						
-MemoryTest-	MemoryTest					
Select the me	mory devices which will be used to hold the following program sections:					
Instruction:	iocm_cntlr 🛛 💌					
<u>D</u> ata:	docm_cntlr					
<u>S</u> tack/Heap:	docm_cntlr					
Boot section re WARNING If you have pla use a debugg	esides in iocm_ontlr aced the Instruction or Data section of this program in an external memory er, bootloader, or ACE file to initialize memory before you can run this prog	y, you must gram!				
		, <b>G</b>				
<u>M</u> ore Info	< <u>B</u> ack <u>N</u> ext >	<u>C</u> ancel				

Dans cette fenêtre vous allez spécifiez les options de l'éditeur de lien du code de test de la mémoire. Vous avez le choix de mettre le code dans la mémoire OCM que nous avons affecté à l'espace programme, ou bien dans le bloc ram interne ou bien dans la RAM DDR Externe. Ce qui est dangereux puisque nous souhaitons tester cette RAM !

De même les données peuvent être stockées en RAM interne ou externe.

Expliquez la différence entre les blocs OCM et le bloc plb\_ram\_if\_cntrl\_1 ? Quelle incidence aurait la sélection de ce bloc pour stocker le code et les data ?









¢	Base Syster	n Builder - Configure Po	eripheral Test A	pplication	?×	
-	The Peripheral Selftest application includes a simple self test for each periperhal in your system (if such a selftest function exists in the driver the peripheral)					
	-PeripheralTest					
	Select the me	mory devices which will be use	d to hold the followin	g program sections:		
	Instruction:	DDR_SDRAM_32Mx32		~		
	<u>D</u> ata:	DDR_SDRAM_32Mx32		~		
	<u>S</u> tack/Heap:	DDR_SDRAM_32Mx32		~		
	Boot section resides in iocm_cntlr WARNING Due to the expected size of this application, it is highly recommended that you place the Instruction and Data sections in an external memory. You will have to use a debugger, bootloader, or ACE file to initialize the memory.					
	<u>M</u> ore Info		< <u>B</u> ack	<u>N</u> ext> <u>C</u> a	ncel	

Vous paramétrez ici les options de l'éditeur de lien du code de test de l'UART.





Le wizard vous propose alors un résumé du mapping mémoire de notre système.

Processor: PPC 405 Processor clock free Bus clock frequence Debug interface: FF On Chip Memory : 4 Total Off Chip Mem - DDR SDRAM 3	5 quency: 100.000000 MHz y: 100.000000 MHz PGA JTAG 48 KB ory : 64 MB 32Mx32 = 64 MB		
The address maps b editing features of XF	elow have been automatica PS.	ally assigned. You c	an modify them using the
Processor OCM:			
Core Name	Instance Name	Base Addr	High Addr
isbram_if_cntlr	iocm_cntlr	0xFFFFC000	0xFFFFFFF
Processor OCM: Core Name	Instance Name	Base Addr	High Addr
dsbram_if_cntlr	docm_cntlr	0x22800000	0x22803FFF
PLB Bus : PLB_V	/34 Inst. name: plb A	ttached Compon	ents:
Core Name	Instance Name	Base Addr	High Addr
plb2opb_bridge	plb2opb_C_RNG0_BA	0x40600000	0x4060FFFF
plb_ddr	DDR_SDRAM_32Mx3	2 0x0000000	0x03FFFFFF
plb_bram_if_cntlr	plb_bram_if_cntlr_1	0x06000000	0x06003FFF
OPB Bus : OPB_	V20 Inst. name: opb	Attached Compo	nents:
Core Name	Instance Name	Base Addr	High Addr
1 10	BS232 Llart	0×40600000	0x4060FFFF

Un petit commentaire au passage. Vous remarquerez le bloc OCM pour le programme a été placé à l'adresse 0xFFFFC000. En fait après un Reset le PowerPC se branche à l'adresse 0xFFFFFFC. Le wizard met ainsi une mémoire à disposition du PowerPC pour un démarrage à froid.

Comme vous êtes maintenant des experts en utilisation de mémoire dans le FPGA, vous n'êtes pas sans savoir que l'on peut initialiser cette mémoire avec des données à la configuration du FPGA, et donc y mettre notre code... Nous reverrons çà un peu plus loin.

Le mapping proposé par le Wizard pourra être modifié par la suite.





#### 3. Détails du système

Dans la fenêtre *System Assembly*, retrouvez comment notre UARTLITE est raccordée au powerpc.

Depuis le menu *Project* sélectionnez la génération de la vue de diagramme bloque. Inspectez le *Block Diagram* résultant et expliquez comment le PowerPC est raccordé au bloque RAM OCM, à la mémoire plb\_bram, à la RAM DDR.

Le fichier testuart.mhs est la description en mode texte de notre système. La syntaxe étant proche du VHDL vous pourrez aisément comprendre les connexions entre les modules.

Ouvrez le fichier, les premières déclarations représentent les pins de connexions au monde extérieur des netlist internes du design.

Que manque-t-il pour affecter le nom de ces signaux à des pins particuliers du FPGA ?

Dans le fichier *mhs*, retrouvez l'instance de l'UART. On y retrouve la configuration 9600 bauds... On passe également la valeur de la fréquence d'horloge de sorte que le driver C de notre UART pourra calculer la valeur à mettre dans le registre de division d'horloge pour régler le bauderate à 9600 bauds.

Vous pourrez également noter à la fin du fichier qu'un module DCM a été instancier pour générer deux horloges en quadrature de phase pour la RAM DDR (Double Data Rate).

Il y une version de hard pour chaque IP de sorte qu'il est possible de choisir la version d'IP que nous sélectionnons pour notre système.

Cliquez sur le filtre *Ports* de la fenêtre *System Assembly* et retrouver les pins externes du design. Développez l'arborescence de l'UART et expliquez ou sont connectez ses signaux. L'interruption est-elle utilisée ?

Cliquez maintenant sur le filtre *Addresses*. Vous retrouvez le mapping mémoire de notre système. Vous avez la possibilité de le modifier manuellement pour des convenances personnelles. Vous pouvez également cliquez le bouton *Generate Addresses* pour que l'outil affecte automatiquement les adresses. Il est possible d'en verrouiller une partie en cliquant la case Lock du bloque voulue.

ATTENTION : derrière cette interface graphique conviviale se cache le code VHDL permettant de sélectionnez le périphérique voulue quand on est dans sa zone mémoire. Un mapping complexe se traduira par un décodeur complexe et donc de la ressource FPGA.





Ouvrez la fenêtre Software Platform Settings.

Software Platfo	rm Settings				Ε	
Processor Instance:	ppc405_0 💌					
Software Platform	Processor Settings					
OS and Libraries	CPU Driver: cpu_ppc405 🔹 CPU Driver Version: 1.00.a 💌					
Drivers	Processor Parameters:					
Interrupt Handlers	Name	Current Value	Default Value	Туре	Description	
	ppc405_0     FYTRA COMPUS			otring	Extra compiler flags used in PSP and library generation	
	-ARCHIVER	powerpc-eabi-ar	-y powerpc-eabi-ar	string	Archiver used to archive libraries for both BSP generation.	
	- COMPILER	powerpc-eabi-gcc	powerpc-eabi-gcc	string	Compiler used to compile both BSP libraries and application	
	CORE_CLOCK_FF	REQ_HZ 10000000	40000000	int	Core Clock Frequency in Hz	
	OS & Library Setting	s OS Version: 1.00.a	Default soft     OS like fun	ware platfor	rm. Provides basic processor related functions and basic	
	Use Library	Version Descript	ion			
	🔲 xilnet	2.00.a 🛛 🖌 Xilinx Ne	tworking TCP/IP stacl	< library		
	xilmfs	1.00.a 🛛 🖌 Xilinx Me	mory File System			
	xilfile	1.00.a 🛛 🔽 Provides	file system call acces	s to simple s	serial I/O devices and	
	xilfatfs	1.00.a 🛛 🖌 Provides	read/write routines to	access files	s stored on a FAT16,	
	lwip	2.00.a 🛛 🖌 IwiP TCF	VIP Stack library			
	Download ThirdParty OS &	Library Definition Files here				
					OK Cancel	

Vous pouvez paramétrer ici la partie logicielle de notre système. Quel est le compilateur croisé utilisé par l'outil Xilinx ?

Nous n'utilisons pas d'OS, mais pour la faire tourner sous Linux c'est ici qu'il faudrait configurer le système.

Pour rendre votre système communiquant sur Internet vous pourriez ajouter la pile lwip libre de droit. ATTENTION nous n'avons pas mis de contrôleur MAC à notre design.

Pour gérer un système de fichier vous pouvez utiliser la librairie *xlmfs* par exemple.

Dans la partie *OS and Librairies* on retrouve le branchement de Stdin et Stdout sur notre UART.

La partie drivers permet d'affecter la version de driver que nous voulons utiliser dans notre projet. Ainsi quand nous allons lancer la commande *Genrate Libriaires and BSP*, l'outil Xilinx copiera les drivers et les librairies que nous avons sélectionnez dans le sous répertoire ppc405 de notre projet.





Ouvrez le fichier *testuart.mss* et retrouvez en mode texte les informations que vous venez de voir dans les interfaces graphiques.

Vous l'avez compris tout notre système est décrit dans les trois fichiers MHS, MSS et UCF.

## 4. Compilation de la plateforme matérielle

Derrière cette interface graphique se cache du code VHDL ou Verilog qu'il ne nous reste plus qu'à synthétiser pour obtenir un fichier bistream permettant de configurer notre FPGA.

Pour vous en convaincre cliquez avec le bouton droit sur l'UART dans la fenêtre *System Assembly* et visualisez le code source de l'UARTLITE.

Pour générer le fichier bitstream exécutez la commande *Generate Bistream* depuis le menu *Hardware*.

Une fois le process terminé ouvrez le fichier *implementation/xflow.log* et retrouvez les ressources FPGA utilisées par notre design.

Dans les *Report Files* retrouvez les ressources FPGA utilisées par l'UARTLITE.

Nous avons donc maintenant un fichier *testuart.bit* que nous pourrions utiliser pour programmer notre FPGA.

Nous allons d'abord nous intéresser aux codes d'exemples.





# 5. Les applications standalone

Dans la fenêtre Project Information Area, sélectionnez l'onglet Applications.

Project Information Area						
Project	Applications	IP Catalog				
Software Projects						
Add Software Application Project						
- 🎇 Default: ppc405_0_bootloop						
📮 🐕 Project: TestApp_Memory						
B Pr Ex B Cc B He B Pr Ex B Cc B Cc B Cc B Cc B Cc B Cc B Cc	ocessor: ppc405 ecutable: C:\tem priler Options ources aders oject: TestApp ocessor: ppc405 ecutable: C:\tem priler Options purces eaders	_U p\TestApp_Me _ <b>Peripheral</b> _0 p\TestApp_Pe	emory\executable.elf ripheral\executable.elf			

Le projet *TestApp\_Memory* est marqué comme étant actif et comme initialisant la BRAM (RAM interne FPGA).

Pour changer l'état d'un projet, cliquez dessus avec le bouton droit.

Editez les options de compilations du projet. On peut voir que l'on passe à l'éditeur de lien un script. Editez ce script (*TestApp\_Memory\_LinkScr.Id*) et retrouvez où se trouve stocké le code de notre application, qu'elle est l'adresse de boot du powerpc et qu'elle est la taille de la pile.

Depuis un explorateur de fichier, parcourez le répertoire de votre projet. Le répertoire ppc405\_0 doit être vide.

Depuis le menu Software, lancez la génération des librairies et du BSP. Le BSP est le Board Support Package. Il est utile quand on a recours à un OS, en fait il va contenir un fichier xparameters.h qui donne la liste des périphériques avec leur adresse, ainsi que les drivers pour l'OS en question.

Revenez sous votre explorateur. Le répertoire ppc405\_0 contient maintenant des sous répertoires avec les drivers des périphériques que vous avez sélectionné, ainsi que les librairies que vous avez activé.

Ouvrez le fichier, ppc405\_0\inlude\xparameters.h et retrouvez tous nos périphériques.

C. ALAYRAC





#### 6. Compilation des applications

Il ne nous reste plus qu'à compiler le code et puis à le télécharger dans le FPGA.

Depuis le menu *Software*, démarrez la commande *Build All User Appllications*. L'exécutable *executable.elf* sera placé dans le répertoire *Test\_Memory*. Ce code est exécutable sur architecture PowerPC et nous l'avons compilé sur un 686. Vous venez sans le savoir de faire votre première compilation croisée.

En fait l'outil xilinx, ouvre un shell Cygwin et y exécute sous linux une compilation croisée à l'aide de la version powerpc\_eabi de gcc.

Modifiez les options de compilation et ajoutez la génération d'un fichier Map aux options de l'éditeur de lien.

🗇 Set Compiler Options 🛛 🔀					
Compiler Tools: powerpc-eabi-gcc					
Environment Debug and Optimization Paths Advanced					
Other Compiler Options to Append					
These options will be appended to the compiler command line					
Symbol Definitions					
Preprocessor (-Wp)					
Assembler (-Wa)					
Linker (-WI) -Map TestApp_Memory/TestApp_Memory.map					
Note: Multiple options should be separated by spaces. To pass a symbol definition to the linker, you would say "-defsym SYMBOL=VALUE"					
OK Cancel					

ATTENTION : La compilation est faite sous Linux, donc pour un répertoire on met / et non pas \ !!!

Editez ensuite le fichier map généré pour voir comment l'éditeur de lien a placé les sections de codes et de données.





#### 7. Mise à jour du bitstream

Il faut maintenant mettre à jour la mémoire interne de notre FPGA avec le code compilé de l'application.

Pour cela lancez la commande *Update Bitstream* depuis le menu *Device Configuration*.

L'outil génère un fichier *download.bit* à partir du fichier *testuart.bit* dans lequel le contenu de la BRAM est pré configuré avec le code de l'application.

#### 8. Téléchargement du code via le lien JTAG

Nous avons maintenant un fichier de configuration avec notre système, et son code. Nous avons donc presque notre carte complète dans ce code !!!! Heureusement pour les passionnés il reste un peu de hardware.

Raccordez la carte ML403 au secteur, branchez le câble de liaison série. Depuis le PC ouvrez un Hyperterminal configurez à 9600 bauds, 8 bits, 1 Stop, pas de parité et pas de contrôle de flux matériel sur COM1.

Démarrez la carte ML403 et depuis le menu *Device Configuration* de XPS lancez la commande *Download Bitsream*.

Les messages de test doivent apparaître sur la console.

-- Entering main() --Starting MemoryTest for DDR\_SDRAM\_64Mx32: Running 32-bit test...PASSED! Running 16-bit test...PASSED! Starting MemoryTest for plb\_bram\_if\_cntlr\_1: Running 32-bit test...PASSED! Running 16-bit test...PASSED! Running 8-bit test...PASSED! -- Exiting main() --





## 9. Projet Test des périphériques

Reprenez la démarche précédente et démarrez le code de l'application de test des périphériques.

Pour vous aider vous pouvez comme précédemment éditer le script pour l'éditeur de lien...

Que se passe-t-il après téléchargement du bitstream ? Pourquoi ?

Lancer le module Debuger de XIIinx (XMD). Lors du premier lancement XPS vous demande de le paramétrer. Choisissez une détection automatique du lien JTAG. Vous pourrez noter dans la console un message du type :

XMD : Conected to PowerPC target. Processor version No: 0x20011430 Address mapping.....

Starting GDB server for "ppc" target (id=0) at TCP port no 1234

Nous reverrons çà un peu plus loin.

Depuis la console tapez XMD% ls Et oui vous êtes sous linux, en fait sous Cygwin (émulateur Linux sous Windows).

Pour télécharger le code de l'application dans la mémoire DDR tapez la commande : XMD%dow TestApp\_Peripheral/executable.elf 0x0

Cette commande va charger à l'adresse 0x0 (c'est à dire le début de la DDR sur notre système), le code de l'application de test des périphériques.

Pour lancer l'application faites ensuite : XMD%con 0x0

Sur votre session Hyperterminal vous devez voir apparaître le message :

-- Entering main() --

-- Exiting main() --





#### 10. Débuggage

Quand vous avez lancer XMD vous avez en fait lancer un serveur gdb. Le service a été activé sur le port TCP 1234 de votre machine. Ce service peut être utilisé localement depuis la console ou bien à distance depuis une autre machine.

Depuis la console vous pouvez tapez des commandes vous permettant de debugger votre code.

Tapez XMD%help

pour voir la liste des commandes disponibles.

Il existe une interface graphique plus conviviale qui permet d'activer ces commandes. Depuis le menu *Debug* lancer la commande *Launch Software Debugger*.

Si vous avez laissé plusieurs projets actif dans l'environnement XPS vous devez choisir l'application à debugger.

Choisissez le code de test des périphériques. Mettez un point d'arrêt sur la ligne print(« --Entering main() --\r\n ») ;

Démarrez le code à partir de la barre d'outil ou du menu Run.

Le logiciel vous demande sur quel serveur GDB vous souhaitez vous connecter. Par défaut il propose le localhost sur le port 1234, mais vous pourriez très bien ici vous connecter à une machine distante.

Une fois le serveur sélectionné, le code est téléchargé dans la mémoire (via une commande *dow* en tache de fond) puis lancé (via une commande *con* ou *run*).

Vous pouvez exécuter pas à pas et parcourir le code.

ATTENTION il faut pour pouvoir debugger tout le code avoir mis l'option de compilation en mode debug (voir les options de compilation du projet).

Vous pouvez également éditer la mémoire vu par le powerPC , par exemple la zone des registres de l'UART.

Retrouvez dans le mapping mémoire l'adresse de notre périphérique RS232. Ouvrez une fenêtre mémoire à cette adresse.

Tapez le caractère U au clavier sous Hyperterminal puis faites un Update Now dans la fenêtre mémoire.

Qu'observez-vous ?

Cliquez à l'adresse 0x40600004 et entrez la valeur 0x56, que voyez-vous apparaître sur votre Hyperterminal.





#### 11. Ajout d'un périphérique

Créez un nouveau projet à partir du Wizard incluant par exemple en plus de l'UART des GPIO.

Reprenez la procédure complète et faites un code de test permettant d'allumer les LED à partir de commandes envoyées depuis le port série.

Prenez le temps de consulter le fichier de contraintes (UCF) les External Ports qui ont été créés...

Pour ajouter un périphérique vous avez également la possibilité d'aller chercher dans le catalogue d'IP l'IP que vous souhaitez ajouter, par exemple nos GPIO. Il faudra la connecter au bus, raccorder ses ports à des blocs voisins ou aux *Externals ports,* penser à compléter le fichier de contraintes, et fixer son adresse dans le mapping mémoire. Cette démarche est toutefois un peu plus lourde et nous ne pourrons pas l'aborder dans le temps qui nous est appartis.

Vous avez maintenant un système opérationnel pour recevoir vos codes standalones ou même un OS, à suivre dans un prochain TD.