

- PROJET RTEL4I -



**Conception détaillée du portage de PREEMPT-RT sur
processeur Microblaze (L2.3-b)**



SOMMAIRE

1. Présentation du processeur softcore MicroBlaze de Xilinx.....	5
2. Portage de PREEMPT-RT sur le processeur softcore MicroBlaze.....	9
2.1. Principes du portage.....	9
2.2. Fichiers sources Linux impactés par le portage PREEMPT-RT.....	9
3. Mise en œuvre de PREEMPT-RT sur le processeur softcore MicroBlaze.....	10
3.1. Carte cible.....	10
3.2. Configuration matérielle : construction du système SoPC.....	11
3.3. Installation des outils Xilinx sous Linux.....	29
3.4. Configuration logicielle : configuration et compilation de Linux pour MicroBlaze.....	31
3.5. Configuration logicielle : configuration et compilation de PREEMPT-RT pour MicroBlaze....	35
4. Mesures des performances.....	40
4.1. Noyau Linux ordinaire.....	40
4.2. Noyau Linux PREEMPT-RT.....	42
5. Conclusion.....	47
6. Références documentaires.....	48
7. Références logicielles.....	48
8. Annexe 1 : fichiers impactés par le portage de PREEMPT-RT pour l'architecture MicroBlaze	49
.....	49
8.1. Liste des fichiers génériques modifiés.....	49
8.2. Liste des fichiers spécifiques modifiés pour l'architecture microblaze.....	57
9. Annexe 2 : shell script goperf-ml403.....	59
10. Annexe 3 : shell script gotraite.....	60
11. Annexe 4 : shell script golatency.....	61
12. Annexe 5 : shell script gohisto.....	62
13. Annexe 6 : shell script make_hist.sh.....	63

Fiche synthétique

Matériel	
Processeur cible	MicroBlaze avec MMU
Fondeur	Xilinx
Outils de synthèse	ISE version 11.4 et supérieur
Carte cible	Xilinx ML403
Logiciel	
Distribution Linux	Fedora 12
Version Linux pour MicroBlaze	Linux 2.6.31
Version compilateur croisé pour MicroBlaze	
Version patch pour MicroBlaze	preempt-rt-2.6.31.12-rt21-microblaze-1.0-00.patch
Version patch PREEMPT-RT pour Linux	2.6.31.12-rt21

Gestion du document :

Date	Version	Modifications	Relecture	Commentaires
25/06/10	1.0	PK	PK	Création fichier

1. PRÉSENTATION DU PROCESSEUR SOFTCORE MICROBLAZE DE XILINX

Le processeur MicroBlaze de Xilinx est un processeur RISC *softcore* entièrement synchrone, son architecture interne étant de type Harvard. Il est optimisé pour une implantation dans un circuit programmable FPGA Xilinx.

La figure 1 donne l'architecture du processeur MicroBlaze

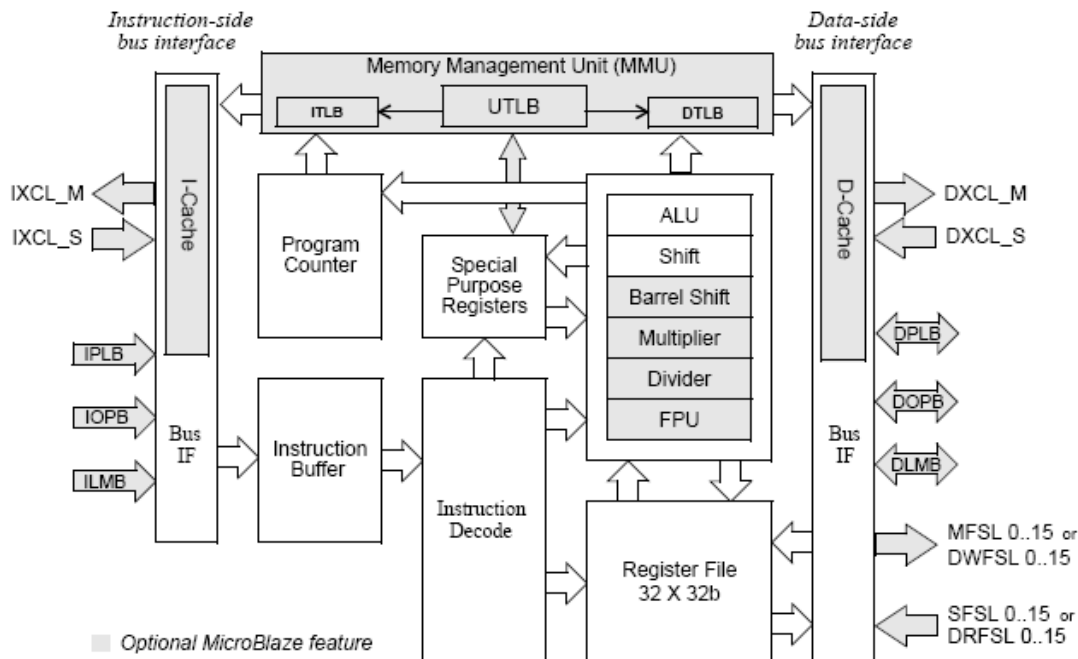


Figure 1 : Architecture du processeur MicroBlaze

Les caractéristiques générales du processeur MicroBlaze sont les suivantes :

- L'architecture du MicroBlaze utilise une architecture Harvard avec ses bus de programme et bus de données séparés.
- Le processeur utilise un jeu d'instruction réduit RISC (*Reduced Instruction Set Computer*).
- Le processeur possède un *pipeline* de 5 étages et de profondeur 3. Trois instructions sont exécutées en 9 cycles d'horloge.
- Les instructions sont de 32 bits avec 3 opérandes et 2 modes d'adressage.
- Le bus d'adresse est de 32 bits.
- Il possède 32 registres de 32 bits à usage général.
- Depuis la version 7 (version actuelle 7.2), le processeur a une unité de gestion de mémoire virtuelle MMU (*Memory Management Unit*) optionnelle.

Les parties en couleur grise sont optionnelles et montrent la configurabilité du processeur MicroBlaze.

Les options matérielles *Barrel Shifter*, *Multiplier*, *Divider* et *FPU* (*Floating Point Unit*) permettent d'accélérer les traitements en implantant par matériel des opérations mathématiques (division, multiplication, opération sur des nombres réels).

Le *pipeline* se compose de 5 étages d'exécution :

1. Lecture de l'instruction dans le buffer d'instruction.
2. Décodage de l'instruction dans le décodeur d'instruction.
3. Accès mémoire pour la lecture des opérandes.
4. Calcul ou exécution de l'instruction dans l'ALU.
5. Ecriture du résultat dans la mémoire.

La figure 2 précise les différentes options de configuration du processeur MicroBlaze en fonction de sa version. De manière générale, il est important d'utiliser la dernière version des outils Xilinx pour avoir accès à la dernière version du processeur MicroBlaze.

Feature	MicroBlaze Versions					
	v4.00	v5.00	v6.00	v7.00	v7.10	v7.20
Version Status	obsolete	obsolete	obsolete	obsolete	deprecated	preferred
Processor pipeline depth	3	5	3/5	3/5	3/5	3/5
On-chip Peripheral Bus (OPB) data side interface	option	option	option	option	option	option
On-chip Peripheral Bus (OPB) instruction side interface	option	option	option	option	option	option
Local Memory Bus (LMB) data side interface	option	option	option	option	option	option
Local Memory Bus (LMB) instruction side interface	option	option	option	option	option	option
Hardware barrel shifter	option	option	option	option	option	option
Hardware divider	option	option	option	option	option	option
Hardware debug logic	option	option	option	option	option	option
Fast Simplex Link (FSL) interfaces	0-7	0-7	0-7	0-15	0-15	0-15
Machine status set and clear instructions	option	Yes	option	option	option	option
Instruction cache over IOPB interface	option	No	No	No	No	No
Data cache over DOPB interface	option	No	No	No	No	No
Instruction cache over Cache Link (IXCL) interface	option	option	option	option	option	option
Data cache over Cache Link (DXCL) interface	option	option	option	option	option	option
4 or 8-word cache line on XCL	4	option	option	option	option	option
Hardware exception support	option	option	option	option	option	option
Pattern compare instructions	option	Yes	option	option	option	option
Floating point unit (FPU)	option	option	option	option	option	option
Disable hardware multiplier ¹	option	option	option	option	option	option
Hardware debug readable ESR and EAR	Yes	Yes	Yes	Yes	Yes	Yes
Processor Version Register (PVR)	-	option	option	option	option	option
Area or speed optimized	-	-	option	option	option	option
Hardware multiplier 64-bit result	-	-	option	option	option	option
LUT cache memory	-	-	option	option	option	option
Processor Local Bus (PLB) data side interface	-	-	-	option	option	option

Figure 2 : Options de configuration du processeur MicroBlaze

La table suivante résume les caractéristiques du processeur MicroBlaze.

MicroBlaze
<ul style="list-style-type: none"> ■ Architecture RISC pipeline ■ Instructions 32 bits ■ Liste de registres fixe ■ Données sur 32 bits ■ 32 niveaux d'interruption ■ Cache d'instructions et de données ■ MMU, FPU, div HW, mul HW

Figure 3 : Caractéristiques du processeur MicroBlaze

La figure suivante présente les performances en DMIPS (*Dhrystons Million Instructions per Second*) de différentes versions du processeur MicroBlaze.

Taille du pipeline	Puissance en DMIPS/MHz
3 étages	0.95 DMIPS/MHz
5 étages	1.19 DMIPS/MHz

Figure 4 : Performances du processeur MicroBlaze

Dans la phase de construction du circuit SoPC (*System on Programmable Chip*) avec l'outil XPS de la suite ISE de Xilinx, il est possible d'inclure différents périphériques standards en utilisant le bus d'adresses/données PLB du processeur MicroBlaze :

- Mémoire.
- Timer.
- Liaison série UART.
- Interface écran LCD.
- E/S parallèles.
- Interfaces Ethernet.
- Interface Compact Flash.
- JTAG.
- ...

2. PORTAGE DE PREEMPT-RT SUR LE PROCESSEUR SOFTCORE MICROBLAZE

2.1. Principes du portage

Le portage de PREEMPT-RT pour le processeur MicroBlaze avec MMU s'est basé sur celui pour le processeur ARM et PowerPC (PREEMPT-RT version 2.6.31.12) et pour le processeur SH (PREEMPT-RT version 2.6.22.1).

Une analyse fine a pu mettre en évidence le rôle des fichiers potentiellement à modifier pour l'architecture MicroBlaze dans le répertoire *arch/microblaze*.

Les principaux fichiers modifiés sont décrits.

Le fichier *arch/microblaze/include/asm/prom.h* est modifié pour accéder de façon atomique à la structure DTS (*Device Tree Structure*) définissant les paramètres à passer au noyau Linux.

Le fichier *arch/microblaze/kernel/entry.S* est modifié pour appeler la fonction du nouvel ordonnanceur PREEMPT-RT *__schedule*.

Le fichier *arch/microblaze/kernel/exceptions.c* est modifié pour un accès atomique à la console en cas d'OOPS.

Le fichier *arch/microblaze/kernel/irq.c* est modifié pour utiliser des spinlocks atomiques.

Le fichier *arch/microblaze/kernel/process.c* est modifié pour appeler le nouvel ordonnanceur PREEMPT-RT *__schedule()*.

Le fichier *arch/microblaze/kernel/prom.c* est modifié pour un accès atomique à la structure DTS.

Le fichier *arch/microblaze/kernel/signal.c* est modifié pour avoir un noyau préemptible.

Le fichier *arch/microblaze/mm/fault.c* est modifié sur le traitement de défaut de page.

Le fichier *drivers/gpio/gpiolib.c* est modifié pour un accès atomique aux E/S GPIO.

Le fichier *drivers/gpio/xilinx_gpio.c* est modifié pour un accès atomique aux E/S GPIO Xilinx.

Le fichier *drivers/i2c/i2c-dev.c* est modifié pour un accès atomique au bus I2C.

Le fichier *drivers/i2c/algos/xilinx_iic/i2c-algo-xilinx.c* est modifié pour un accès atomique au bus I2C Xilinx.

Le fichier *drivers/net/xilinx_emaclite.c* est modifié pour un accès atomique à l'interface Ethernet emaclite Xilinx.

Le fichier *drivers/serial/uartlite.c* est modifié pour un accès atomique à l'interface UART uartlite Xilinx.

Le fichier *drivers/spi/spi_bitbang.c* est modifié pour un accès atomique au bus SPI en mode bit bang.

Le fichier *include/asm-generic/bitops/atomic.h* est modifié pour utiliser plutôt *raw_local_irq_xxx()* que *local_irq_xxx()* pour homogénéité avec le patch général PREEMPT-RT.

2.2. Fichiers sources Linux impactés par le portage PREEMPT-RT

La liste des fichiers sources Linux impactés par le portage de PREEMPT-RT sur l'architecture MicroBlaze est donnée en annexe 1.

On retrouve globalement un ensemble de fichiers indépendants de l'architecture pour l'implantation de la préemption Temps Réel mou dans le noyau Linux et un ensemble de fichiers modifiés concernant l'architecture MicroBlaze...

3. MISE EN ŒUVRE DE PREEMPT-RT SUR LE PROCESSEUR SOFTCORE MICROBLAZE

3.1. Carte cible

La carte choisie pour le portage de PREEMPT-RT sur le processeur MicroBlaze est une carte Virtex-4 de Xilinx : carte ML403.

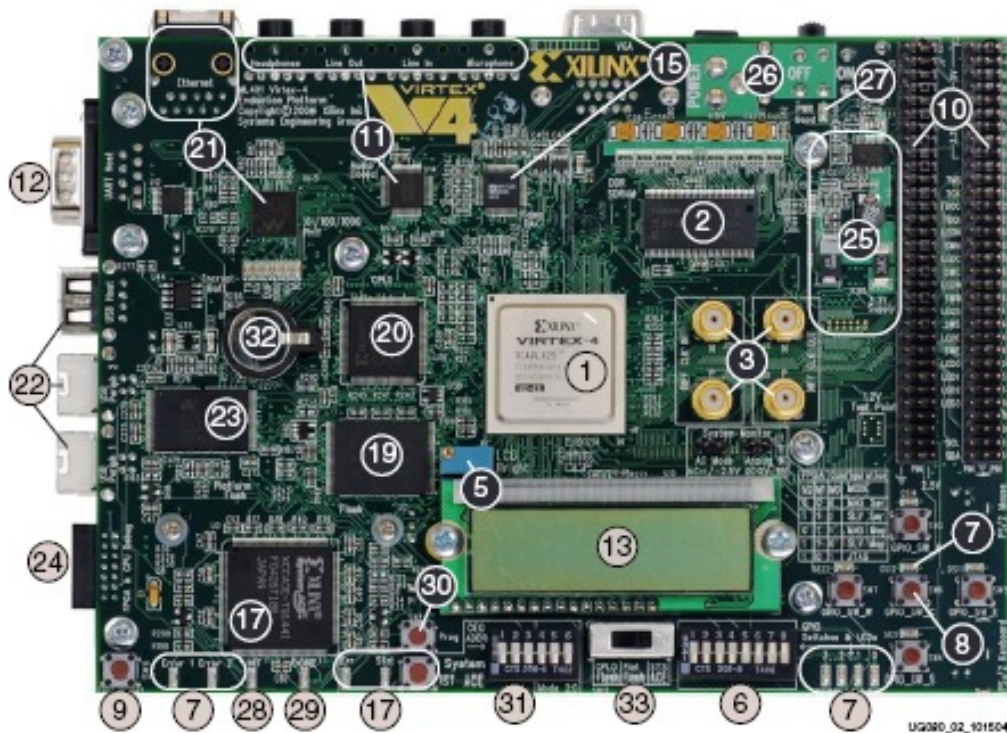


Figure 5 : Carte cible Xilinx ML403

La carte ML403 possède les caractéristiques suivantes :

- Circuit FPGA XC4VFX12-FF668-10.
- 8 Mo de SRAM 32 bits.
- 64 Mo de SDRAM DDR32 bits.
- 8 Mo de mémoire Flash.
- Oscillateur à 100 MHz.
- 1 support CompactFlash type I par interface System ACE.
- 1 interface Ethernet 10/100/1000 Mb/s.
- 1 port série (RS-232 DB9).
- 1 port d'extension pour carte fille maison.
- 1 connecteur JTAG.
- Boutons poussoirs.
- Leds utilisateurs.
- 1 afficheur LCD 2x16 caractères.
- Bus I2C.
- 4Kb d'EEPROM SPI.

- Bus USB.
- Sortie audio AC97.
- Connecteurs PS2 clavier et souris.
- Sortie VGA.

3.2. Configuration matérielle : construction du système SoPC

Il convient à l'aide de l'outil de synthèse XPS de Xilinx de construire le système SoPC compatible avec le portage PREEMPT-RT sur processeur MicroBlaze. A l'issue de la synthèse, on récupérera le fichier *.bit* de programmation du circuit FPGA de la carte cible ML403 ainsi qu'un fichier *.dts* qui décrit le système SoPC en termes de types de périphériques, d'adresses de base et de numéros d'interruption, fichier qui sera utilisé lors de la compilation du noyau Linux.

Le design de référence pour Linux embarqué avec ou sans l'extension Temps Réel a été construit avec l'outil BSB (*Base System Builder*) de XPS.

Ce design standard est une bonne base de travail pour faire tourner dans un premier temps Linux embarqué.

La figure présente le design de référence standard :

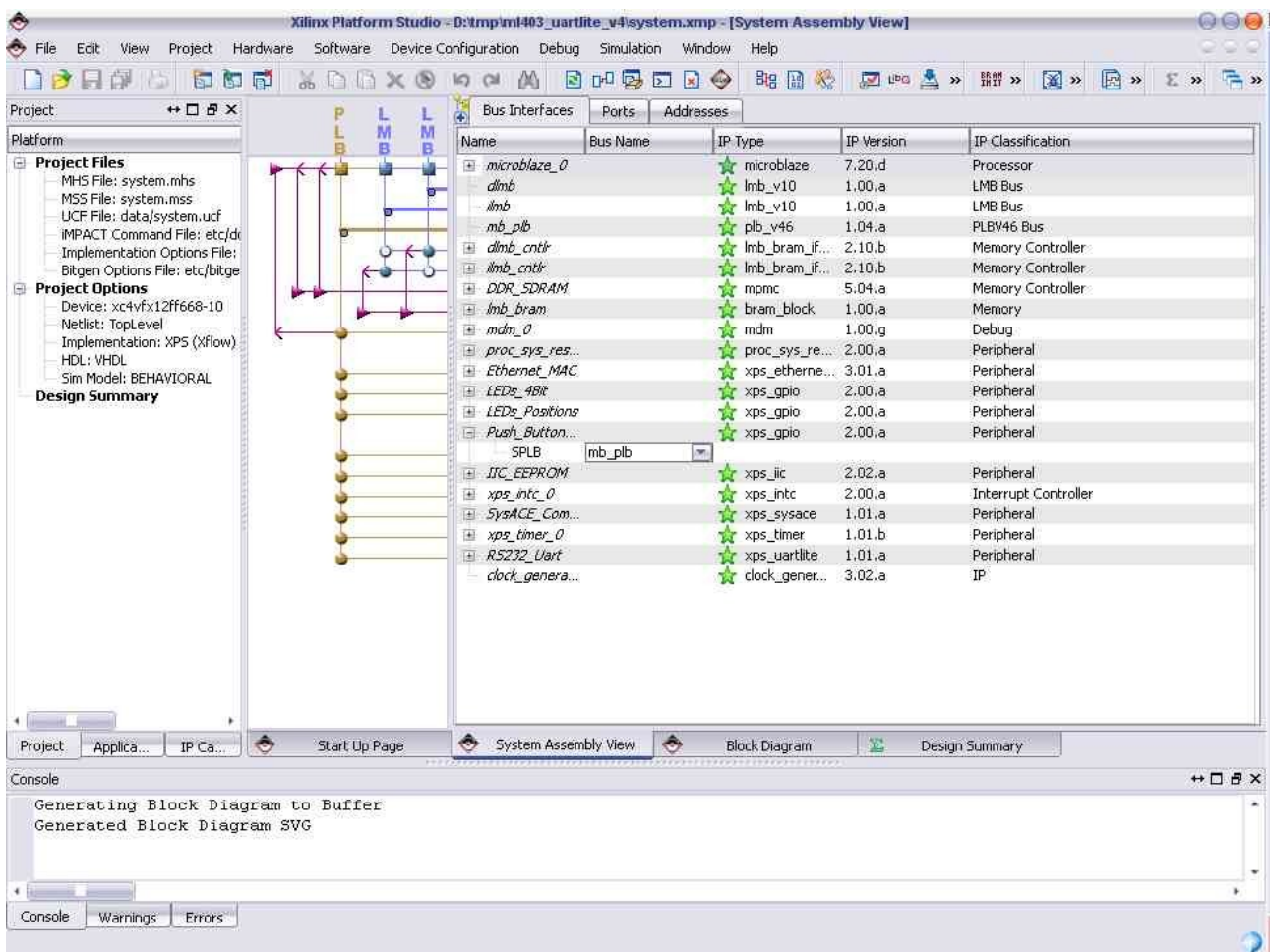


Figure 6 : Design de référence standard XPS de la carte cible Xilinx ML403

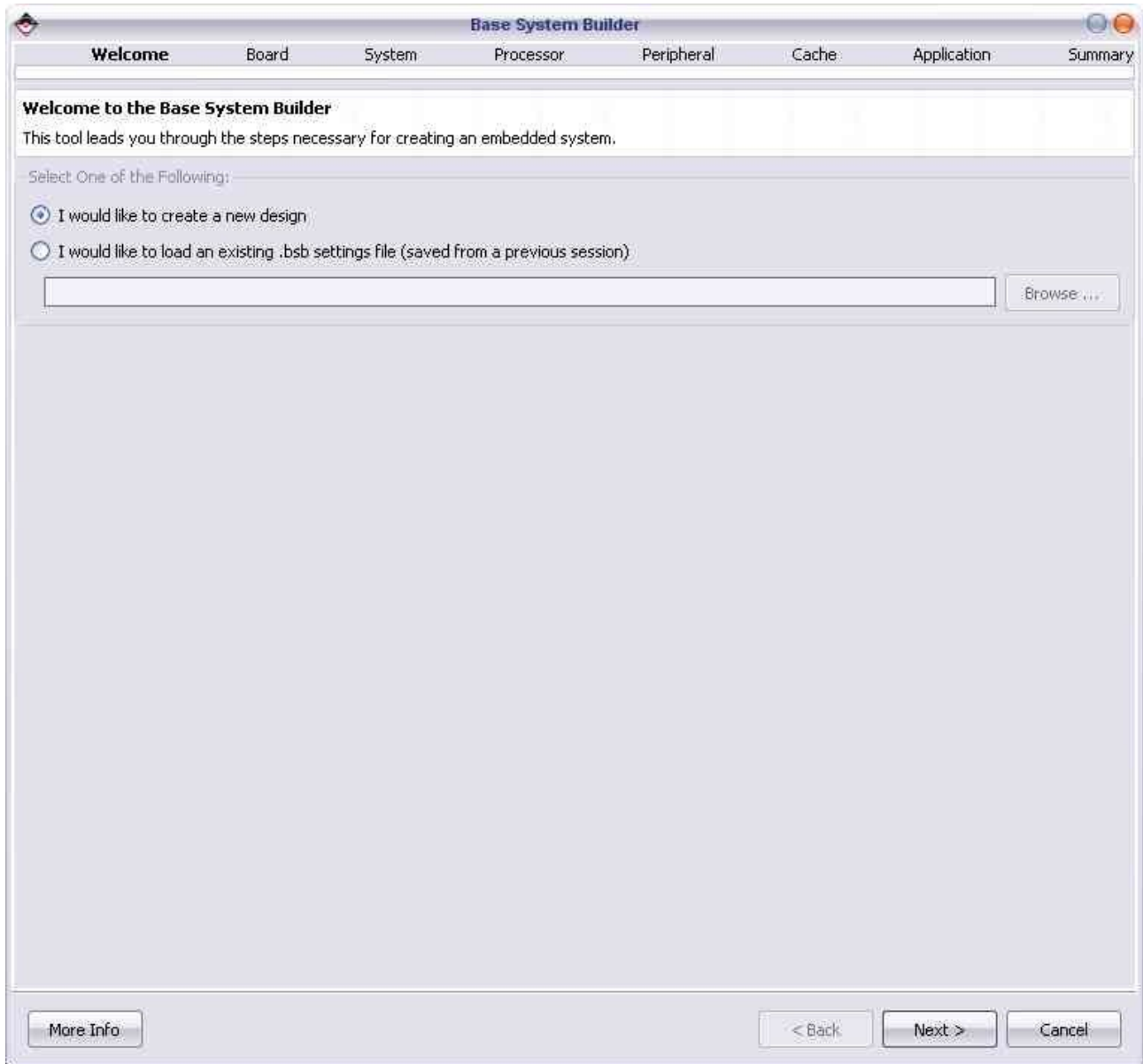
Les figures suivantes montrent la construction du système SoPC.

Pour pouvoir intégrer le patch PREEMPT-RT, le processeur MicroBlaze est configuré avec une MMU (*Memory Management Unit*).

Il convient de créer un projet XPS avec l'outil BSB :



Figure 7 : Création d'un nouveau projet SoPC avec BSB



Figures 8 : Création du projet SoPC

On choisit comme carte cible la carte Xilinx ML403 :

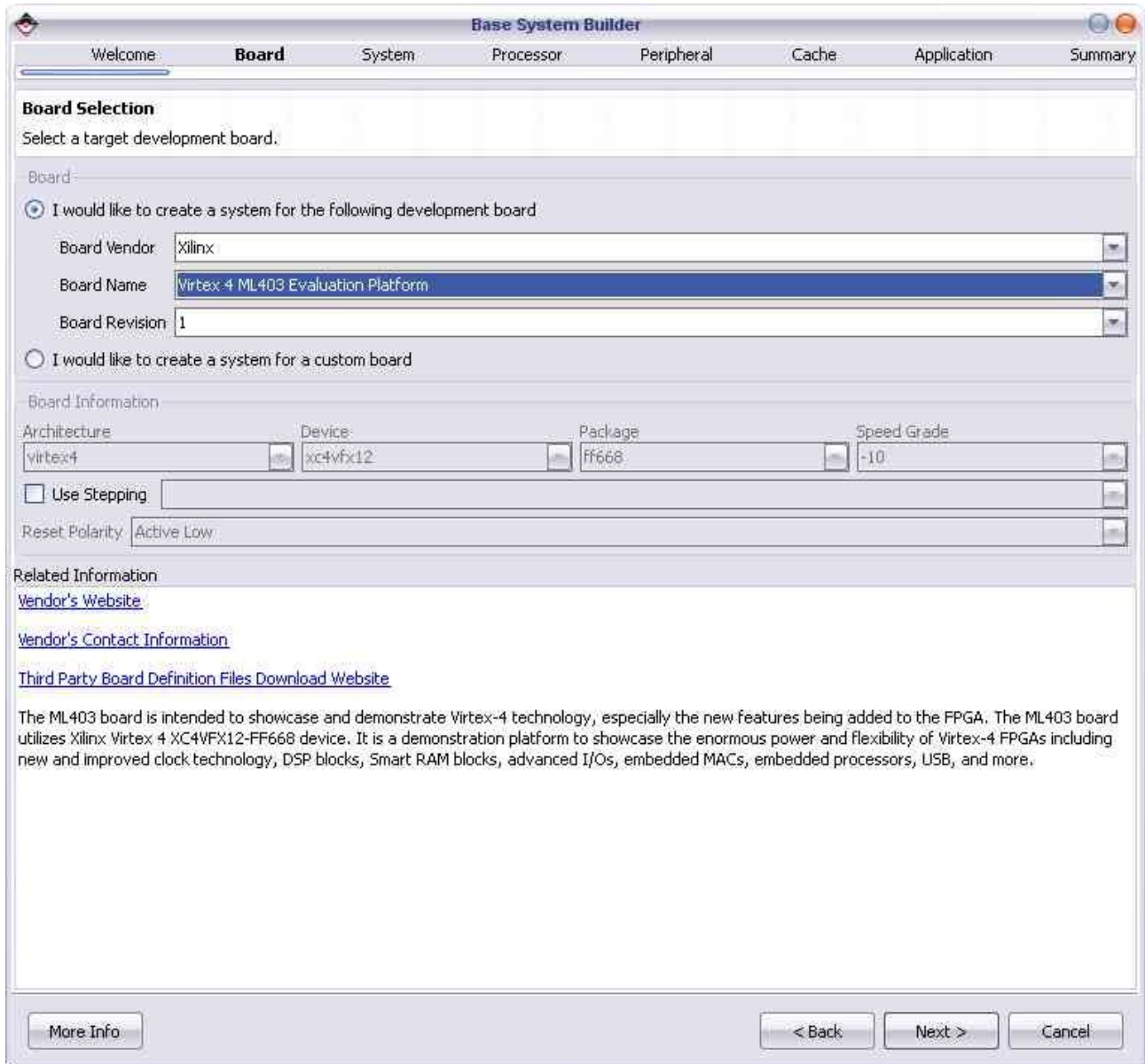


Figure 9 : Choix de la carte cible Xilinx ML403

On construit un système SoPC monoprocesseur :

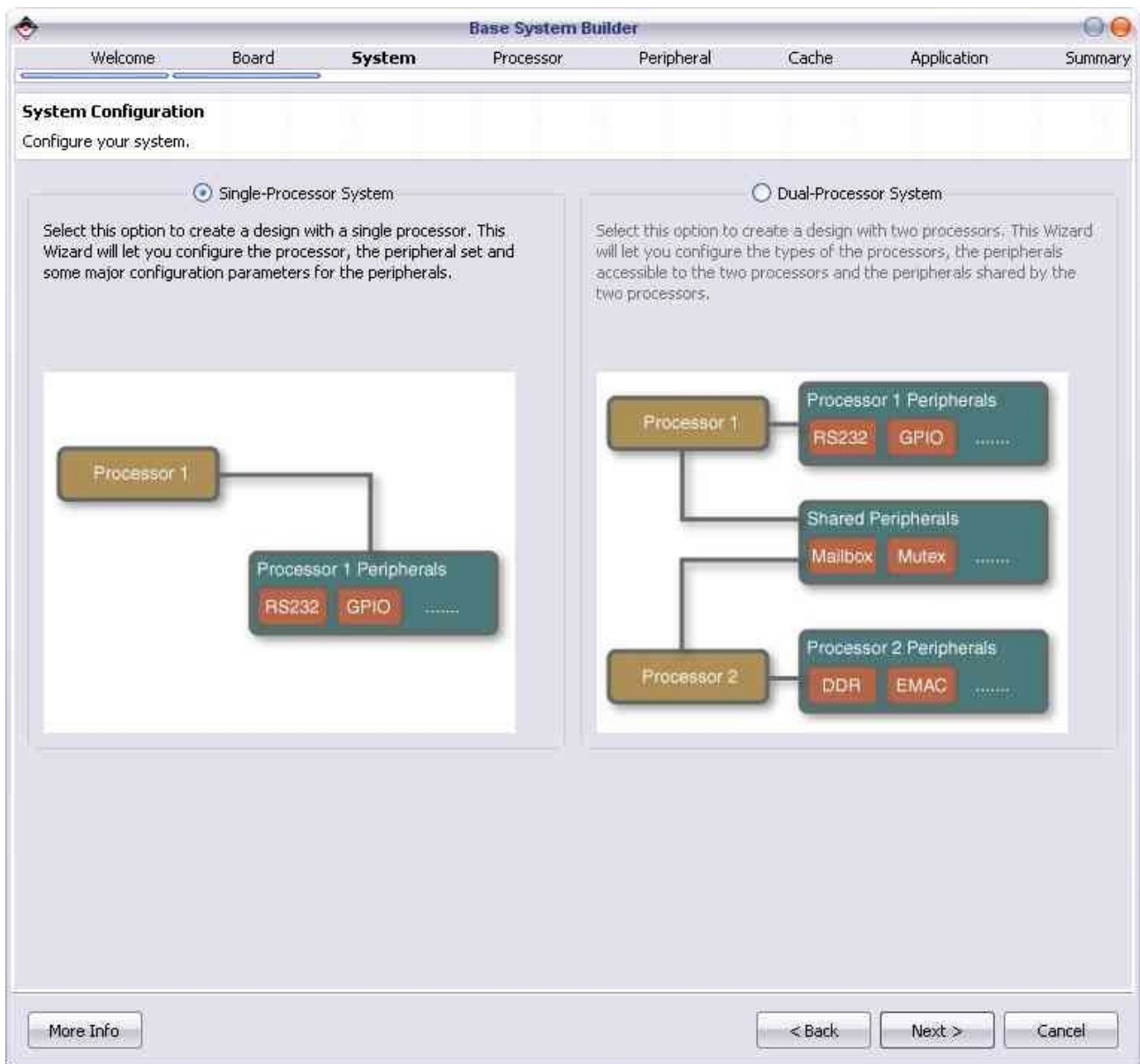


Figure 10 : Construction d'un système SoPC avec un seul processeur

On choisit le processeur MicroBlaze (le circuit FPGA possède un processeur *hardcore* PowerPC) et une horloge système de 100 MHz :

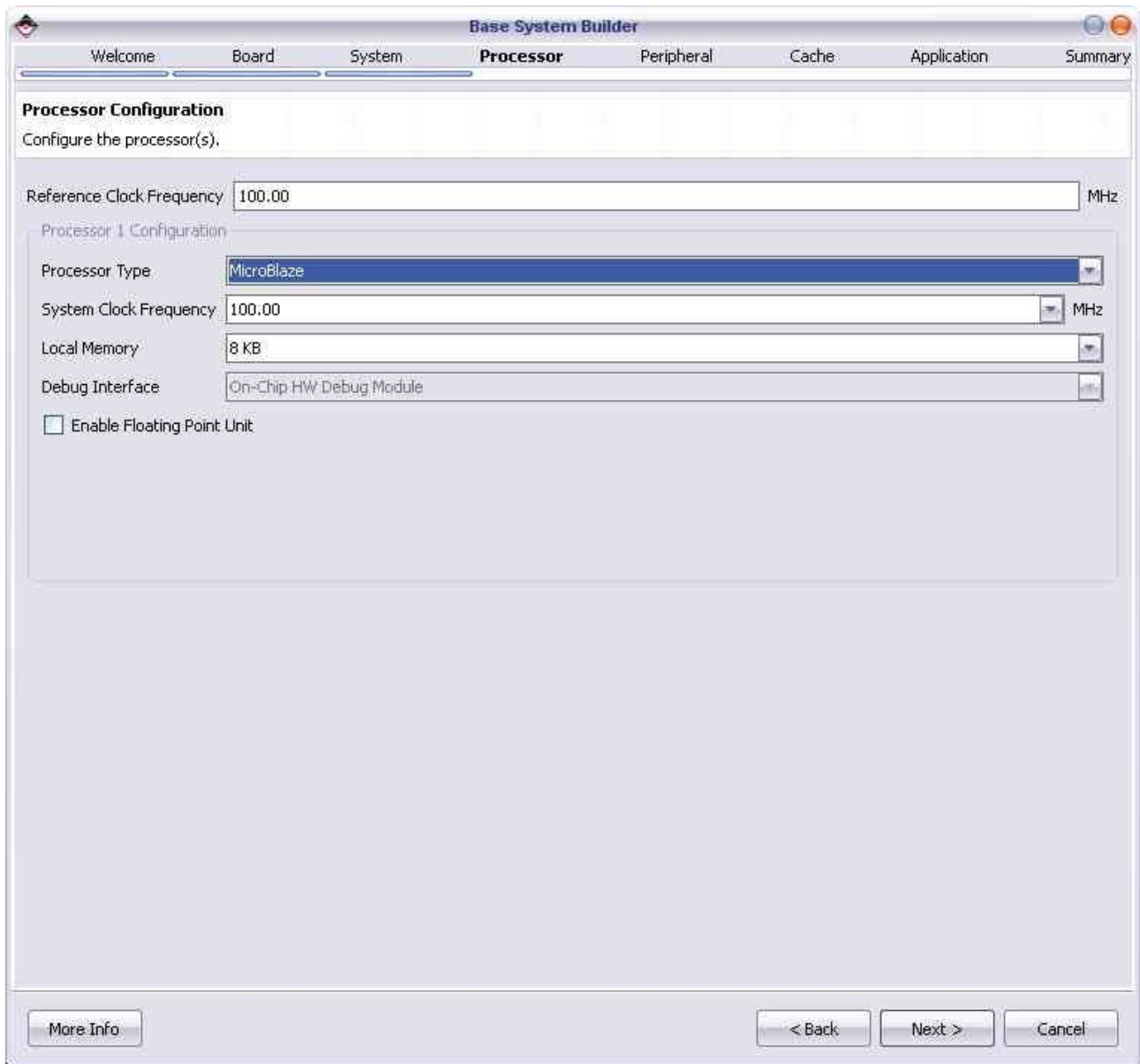


Figure 11 : Choix d'utilisation du processeur MicroBlaze

Un ensemble de périphériques intégrés dans le système SoPC est fourni par défaut en corrélation avec le respect d'usage des broches du circuit FPGA (fichier de contraintes *.ucf*) :

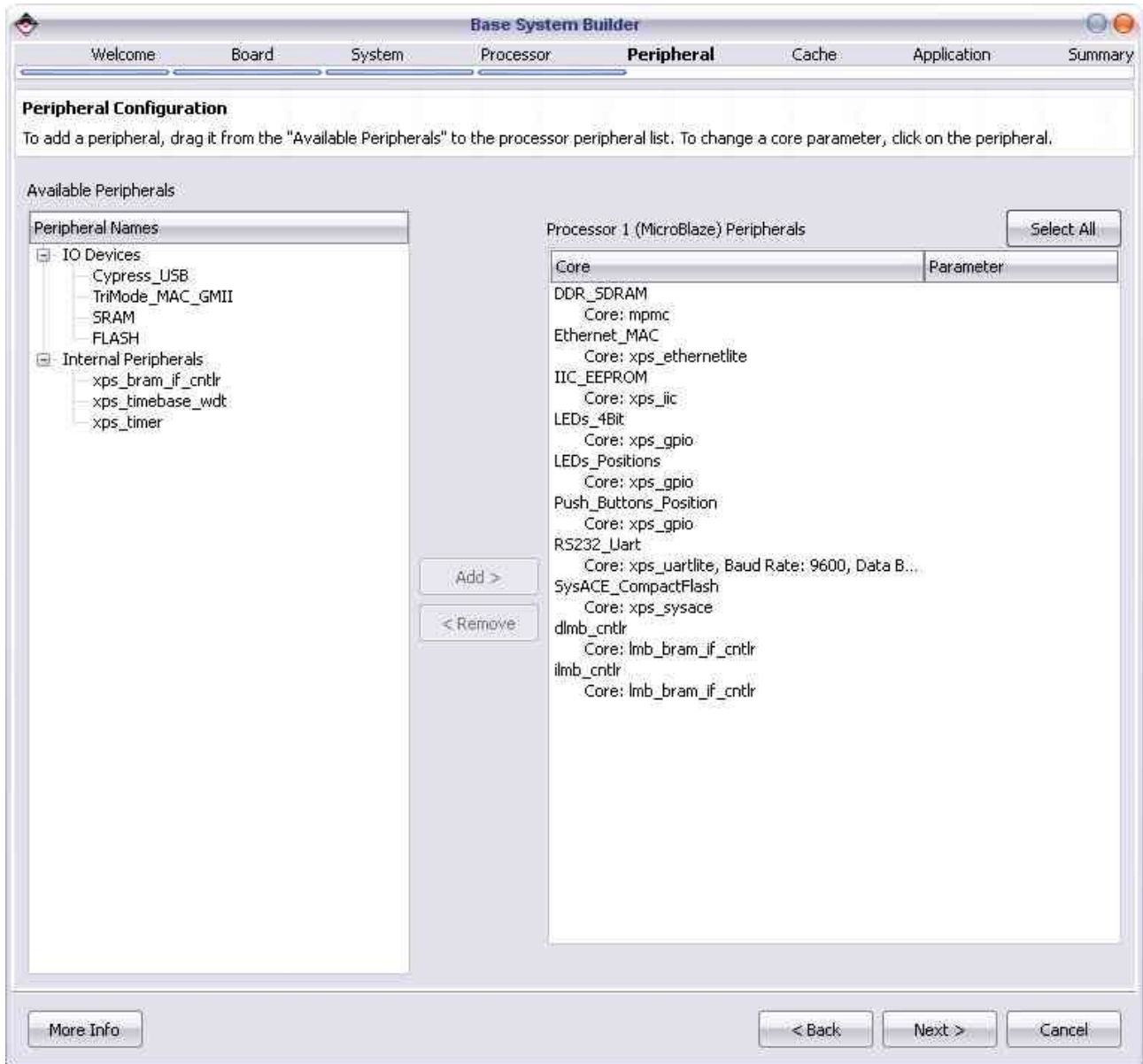


Figure 12 : Création d'un système SoPC par défaut

Pour le support de Linux embarqué, on rajoute le périphérique timer *xps_timer* avec les 2 timers de validés : un timer est utilisé comme *tick timer*, l'autre comme *clock event*.

On validera toutes les interruptions pour les périphériques (case *Use Interrupt* cochée) :

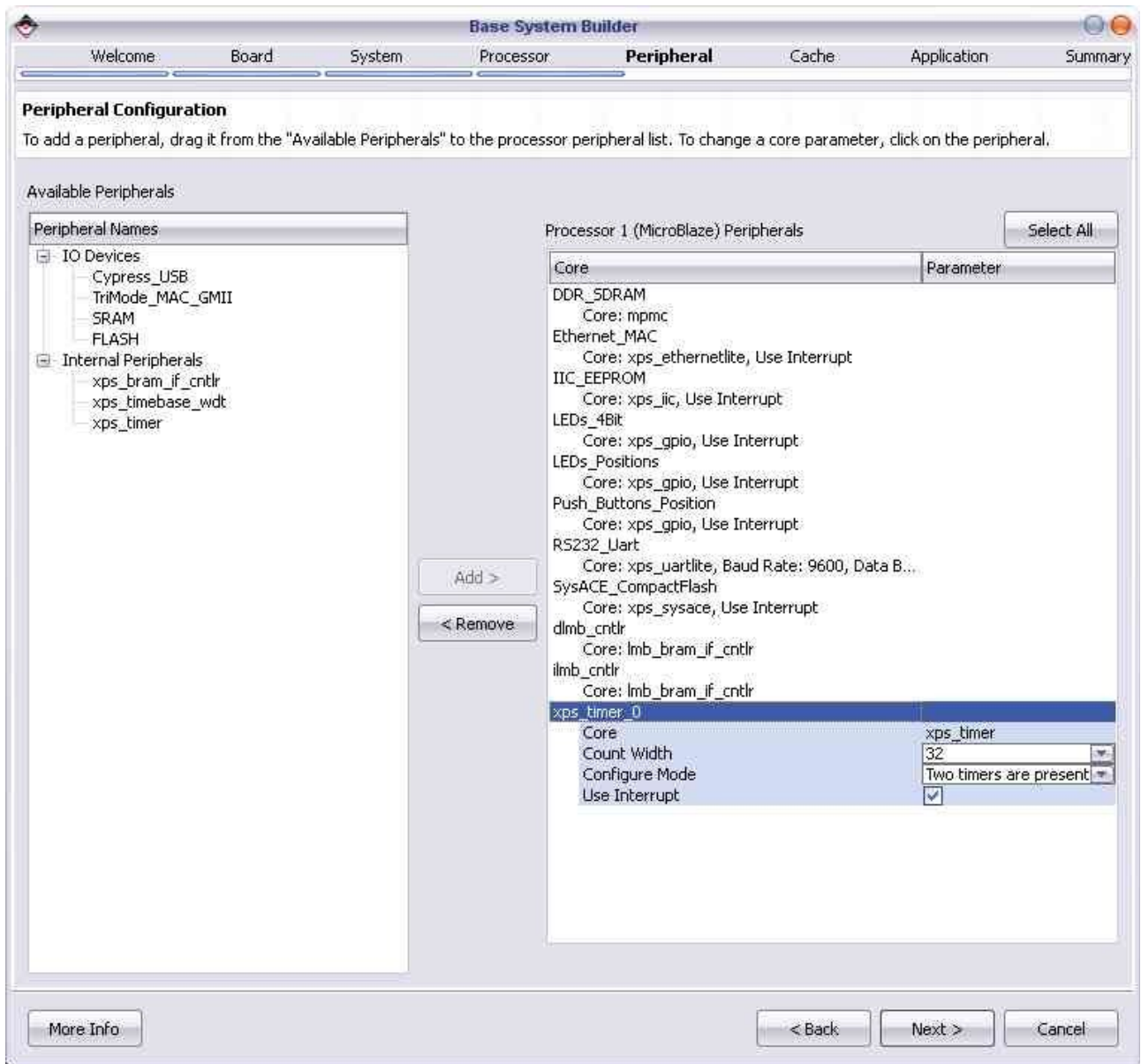


Figure 13 : Rajout du périphérique *xps_timer*

On validera le cache d'instructions et de données :

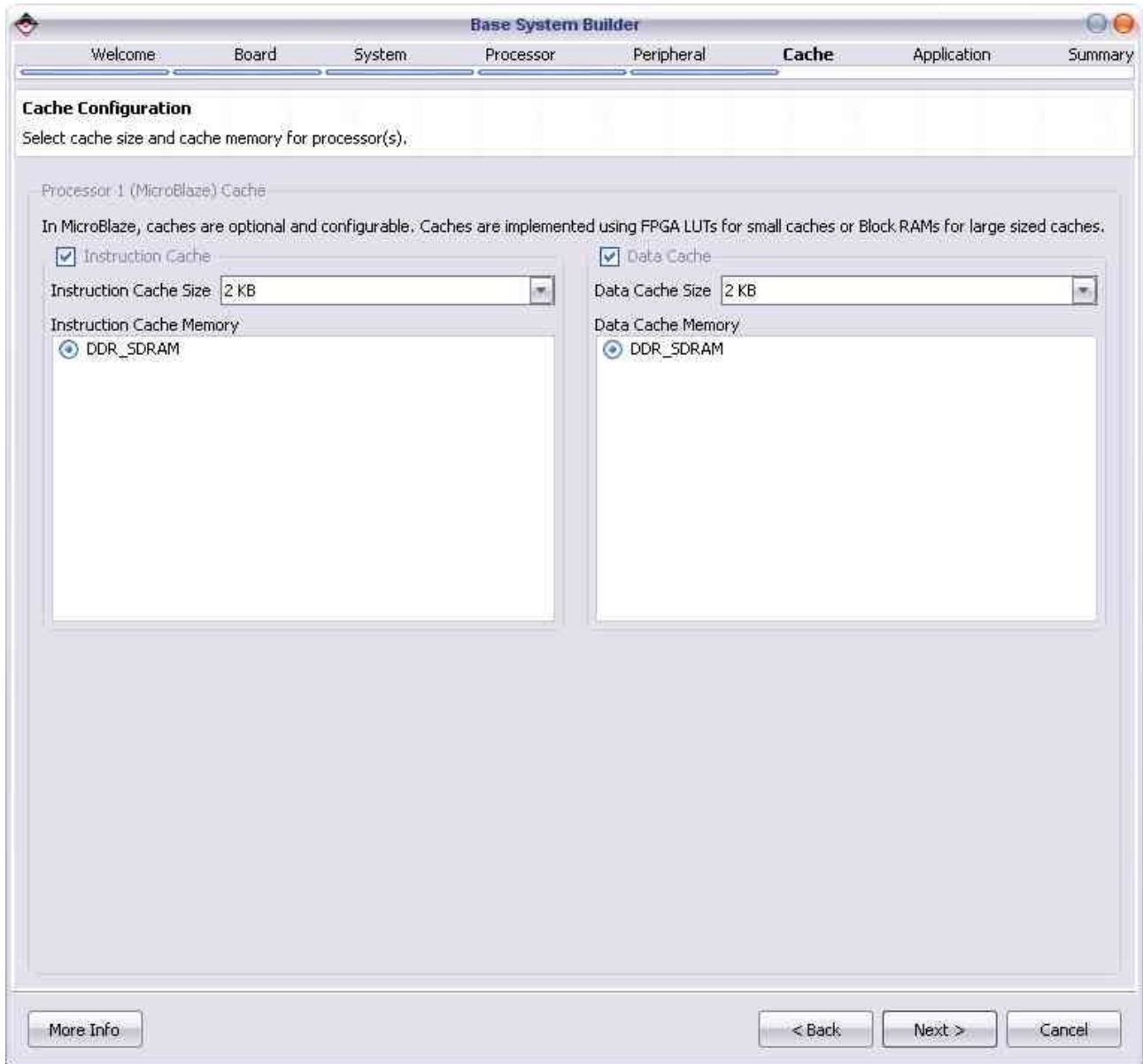


Figure 14 : Validation des caches d'instructions et de données

On laissera la génération des programmes de test :

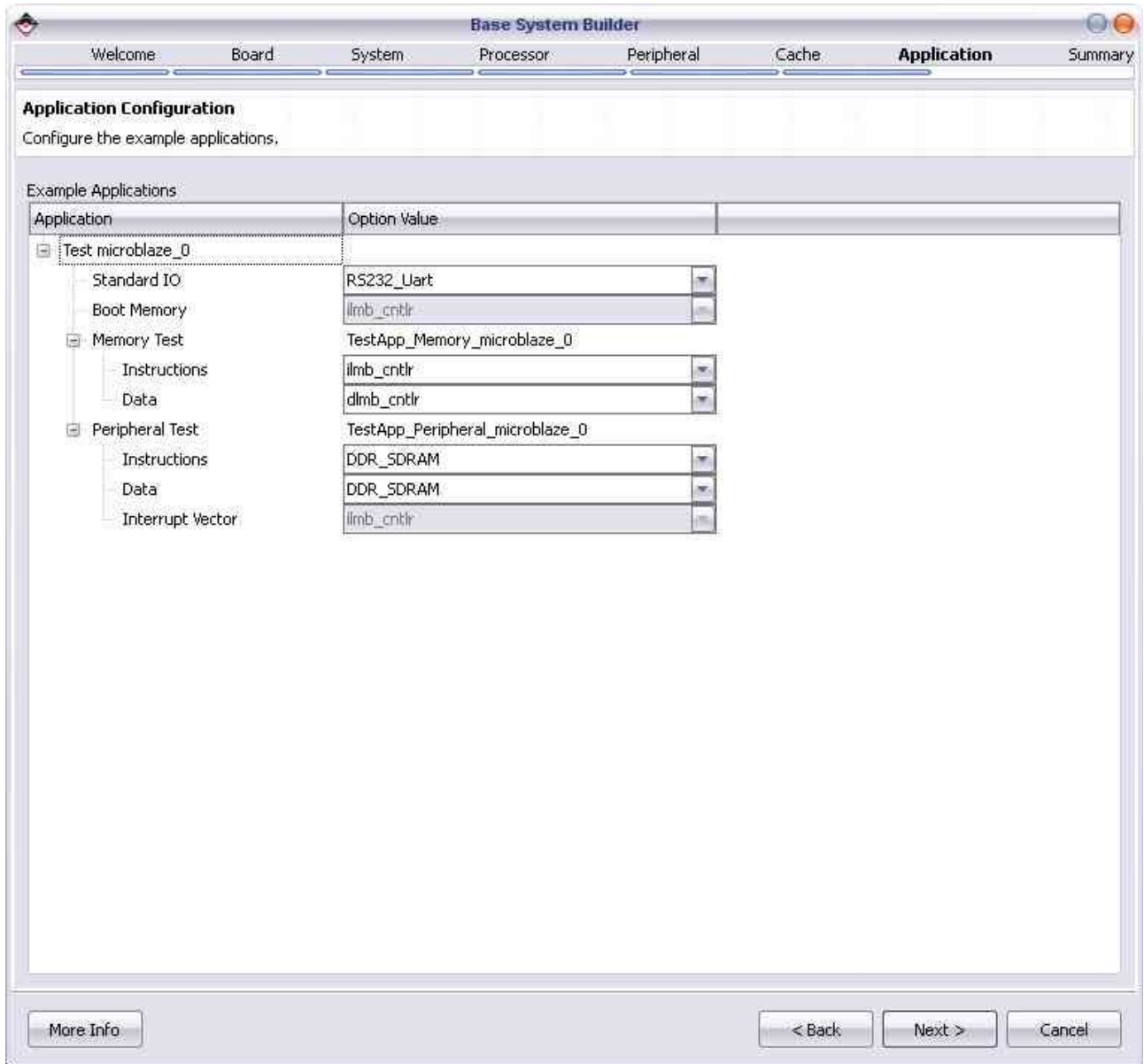


Figure 15 : Génération des programmes de tests

On a alors le récapitulatif du système SoPC généré :

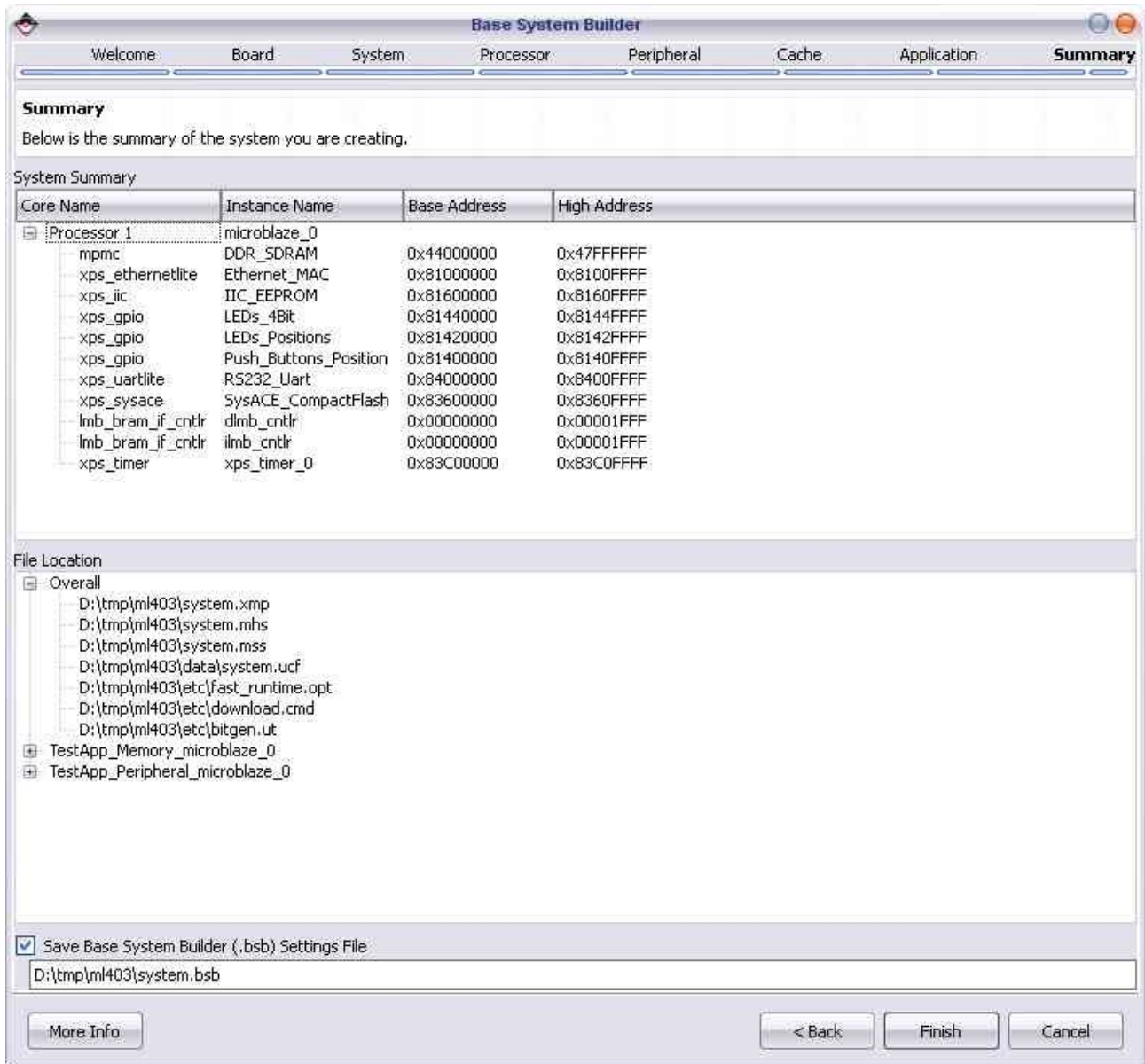


Figure 16 : Récapitulatif du système SoPC généré

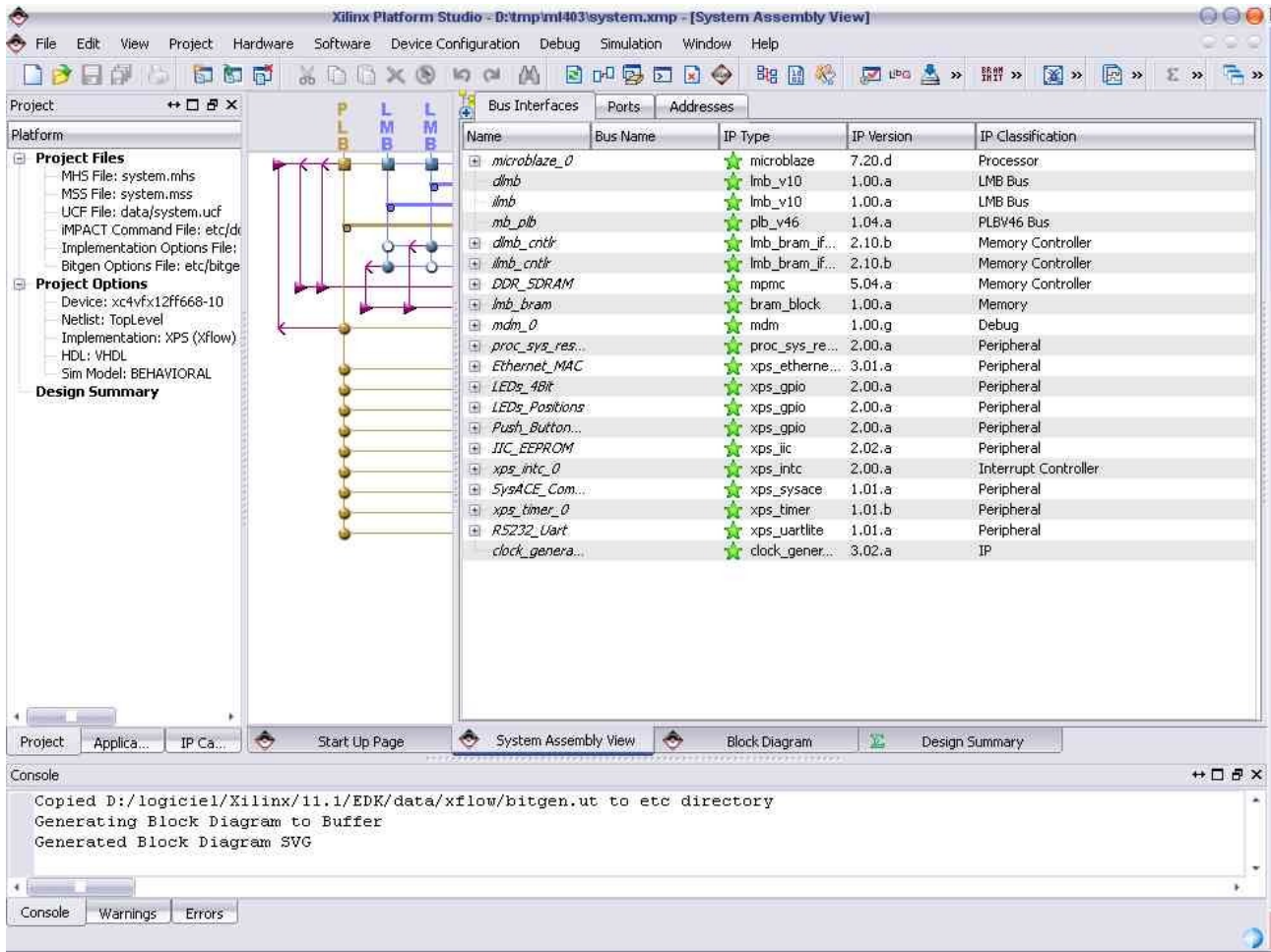


Figure 17 : Système SoPC généré

Il convient ensuite dans XPS de reconfigurer le processeur MicroBlaze pour repréciser les adresses de base des caches d'instructions et de données et de valider la MMU :

- Adresse de début : 0x50000000.
- Adresse de fin : 0x50000000 + taille mémoire SDRAM.

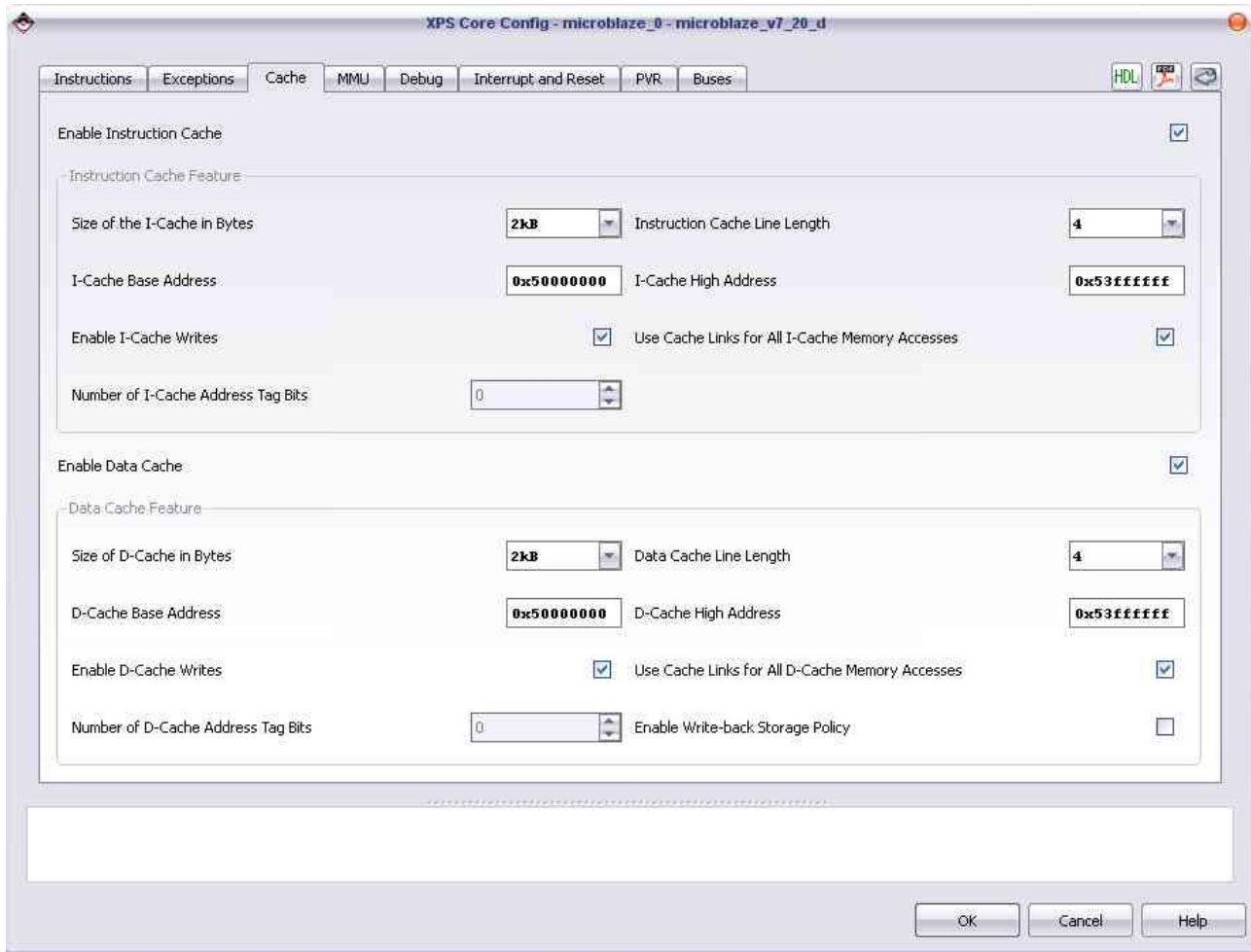


Figure 18 : Reconfiguration des adresses de caches d'instructions et de données du processeur MicroBlaze

La MMU est validée :

- Memory Management : virtual.
- Data Shadow : 4.
- Instruction shadow : 2.
- Enable Access to Memory Management Special Registers : full.
- Number of Memory Protection Zones : 2.

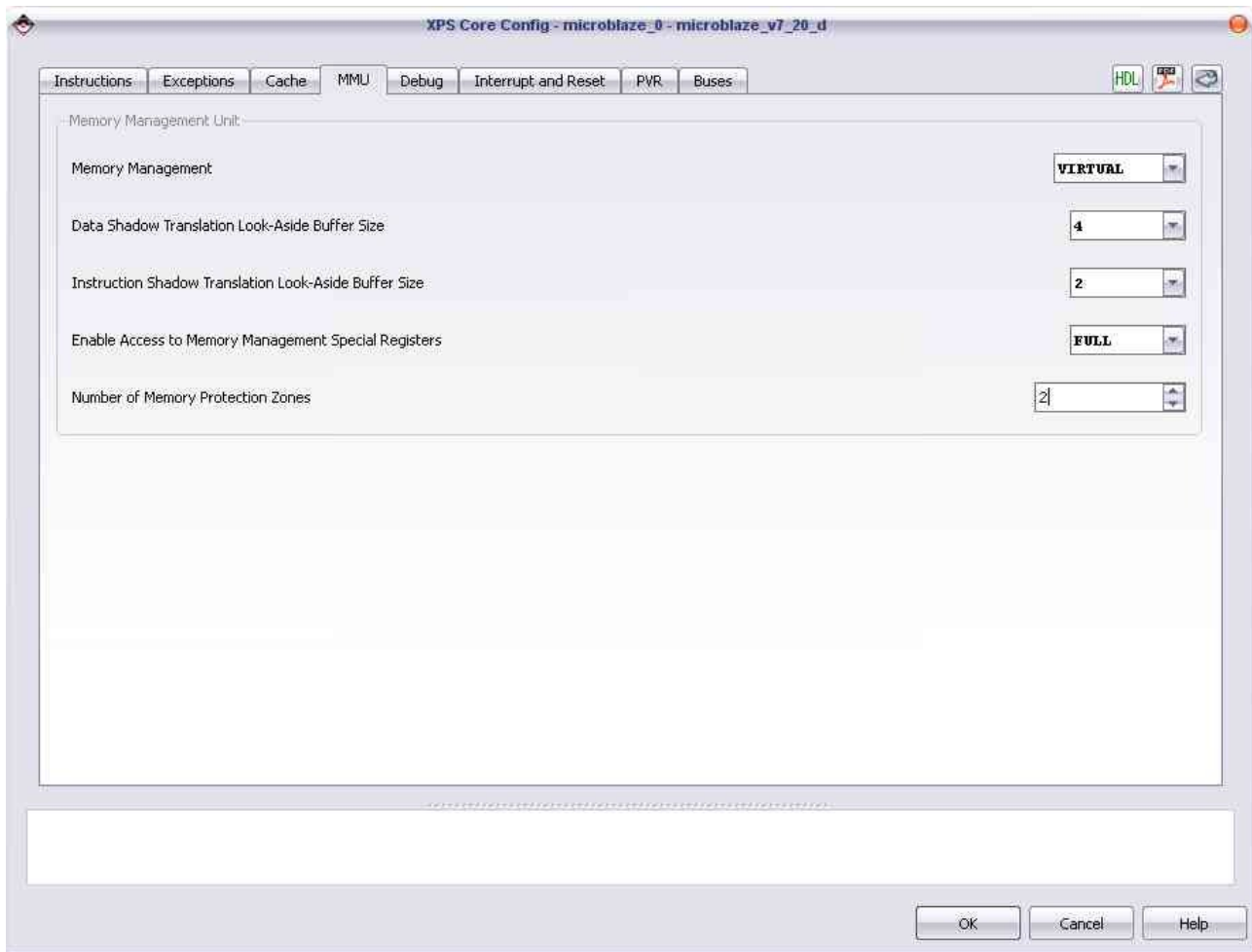


Figure 19 : Validation de la MMU du processeur MicroBlaze

On vérifiera la priorité des interruptions :

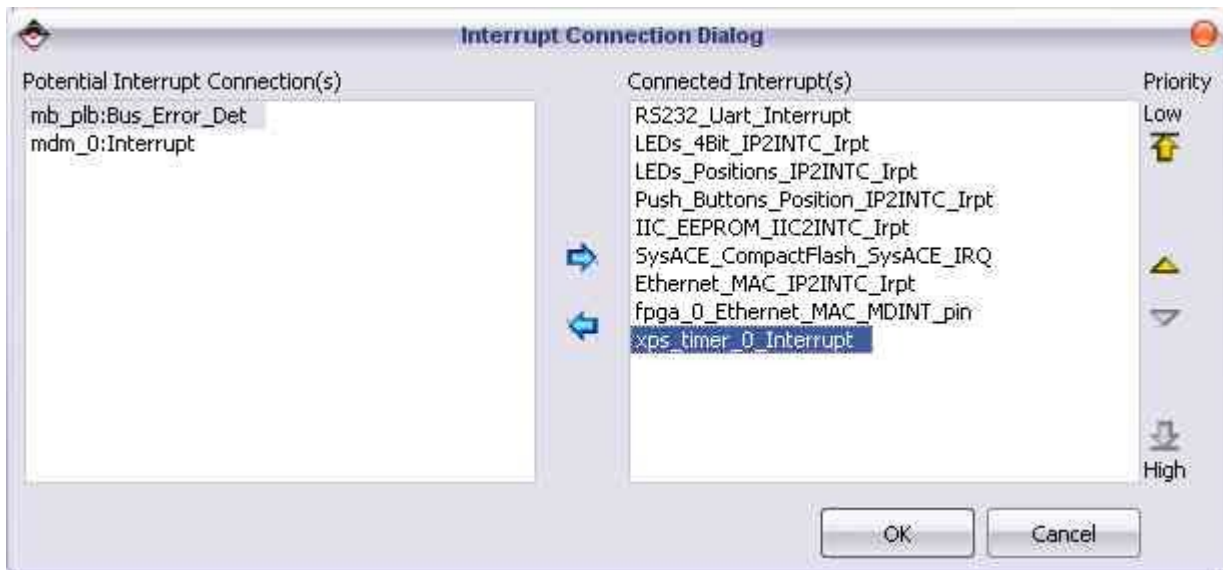


Figure 20 : Configuration des interruptions

On vérifiera les adresses de base. On précisera l'adresse de base de la mémoire SDRAM DDR (0x50000000) :

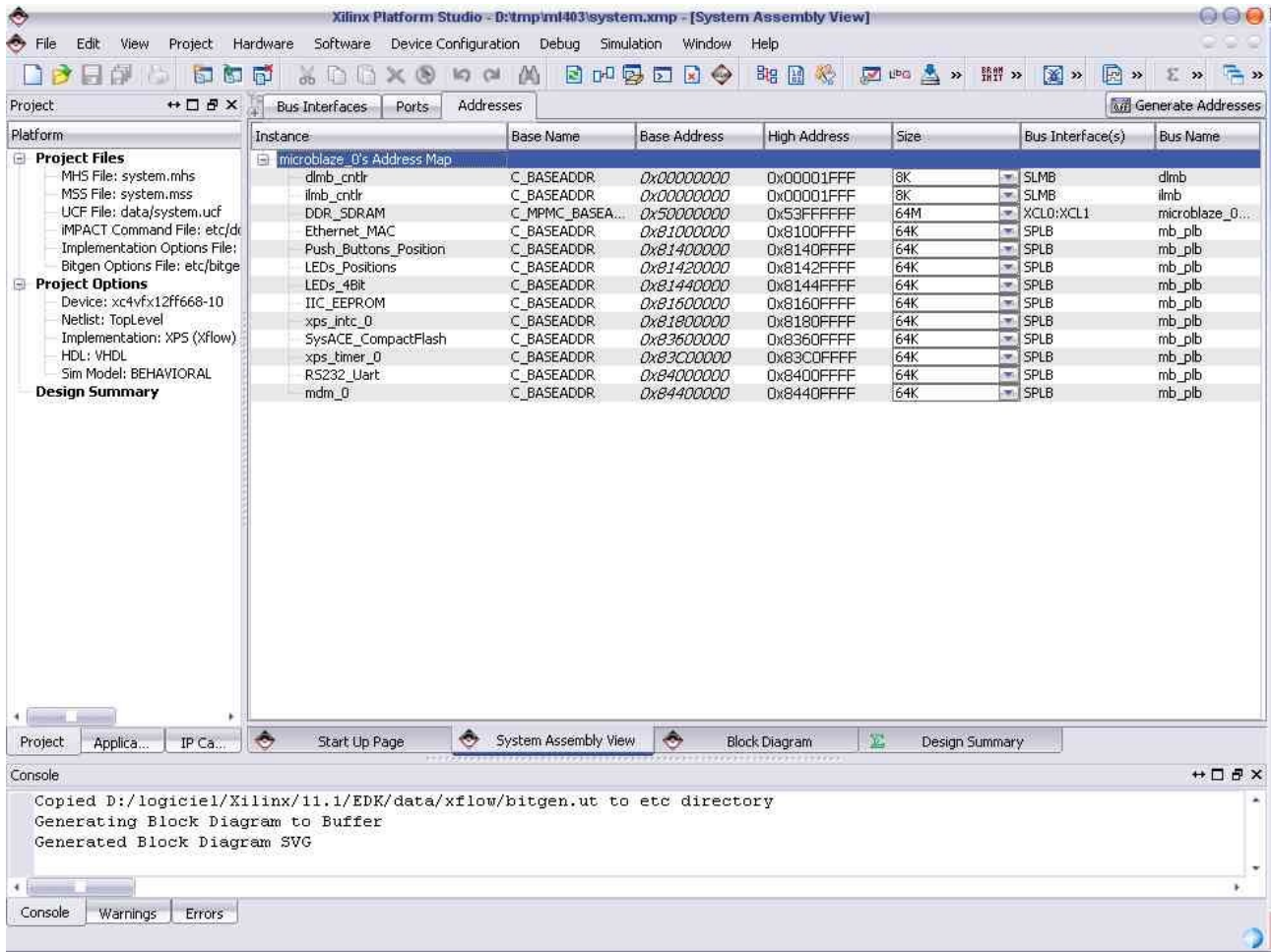


Figure 21 : Configuration de l'adresse de base de la mémoire SDRAM DDR

Le design du système SoPC étant terminé, il convient de générer le fichier *.bit* de programmation du circuit FPGA. Dans le menu *Project*, il faut sélectionner le choix *Export Hardware Design to SDK* et cliquer ensuite sur le bouton *Export Only* :



Figure 22 : Génération du fichier system.bit

Il convient enfin de générer le fichier .dts.

Il faut au préalable installer un plugin spécial et le recopier dans le répertoire du design :

```
$ git clone git://git.xilinx.com/device-tree.git
```

```
$ cd device-tree
```

```
$ cp -r bsp repertoire_du_design
```

On relance XPS. Dans le menu *Software*, il faut sélectionner le choix *Software Platform Settings*. Dans le menu *Software Platform*, il faut valider pour le choix *OS & Library Settings, device-tree* :

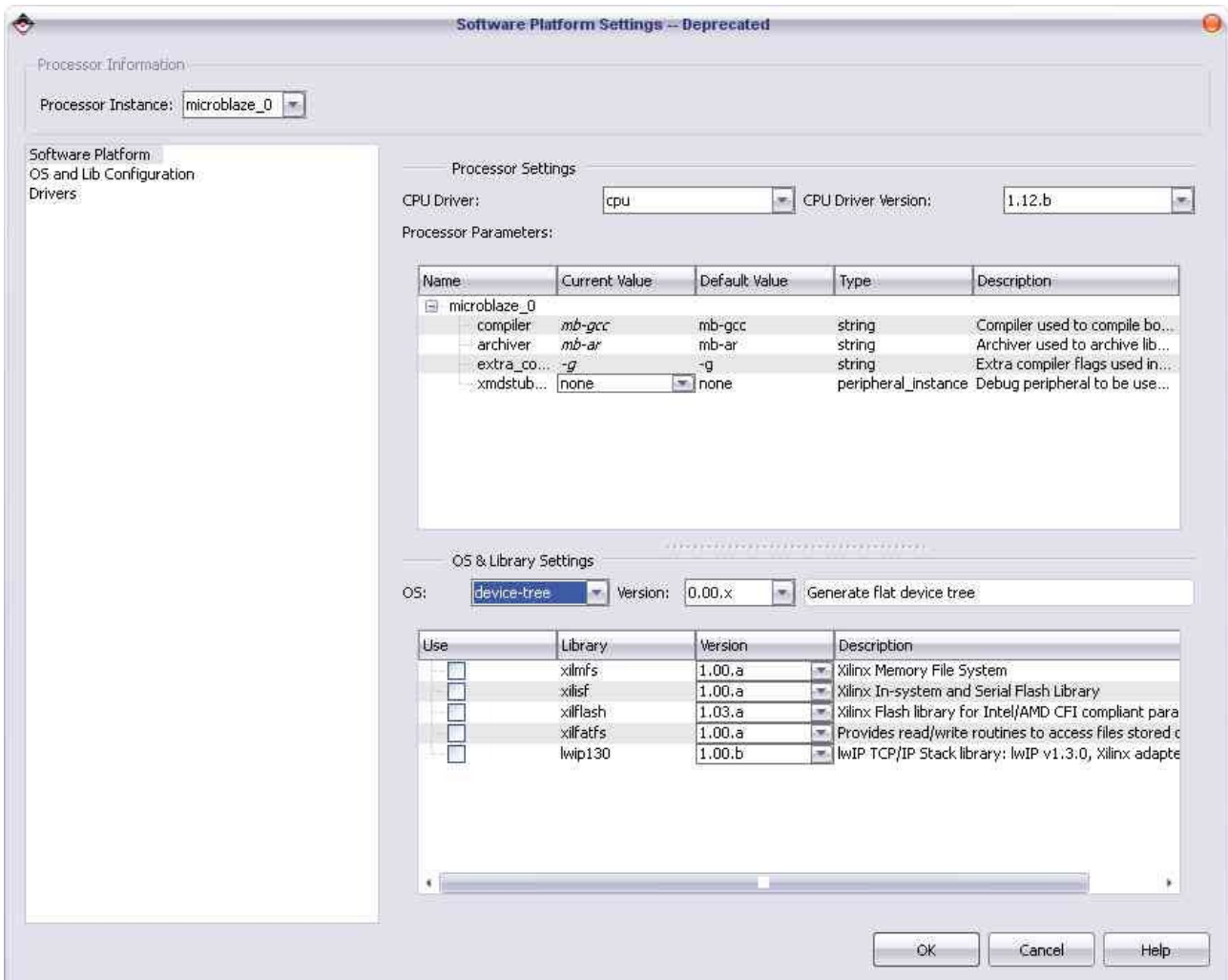


Figure 23 : Configuration pour la génération du fichier *.dts*

Dans le menu *OS and Lib configuration*, il faut préciser la valeur de *bootargs* (pour le passage d'arguments au boot du noyau Linux) soit ici *console=ttyUL0 ip=192.168.0.100* :

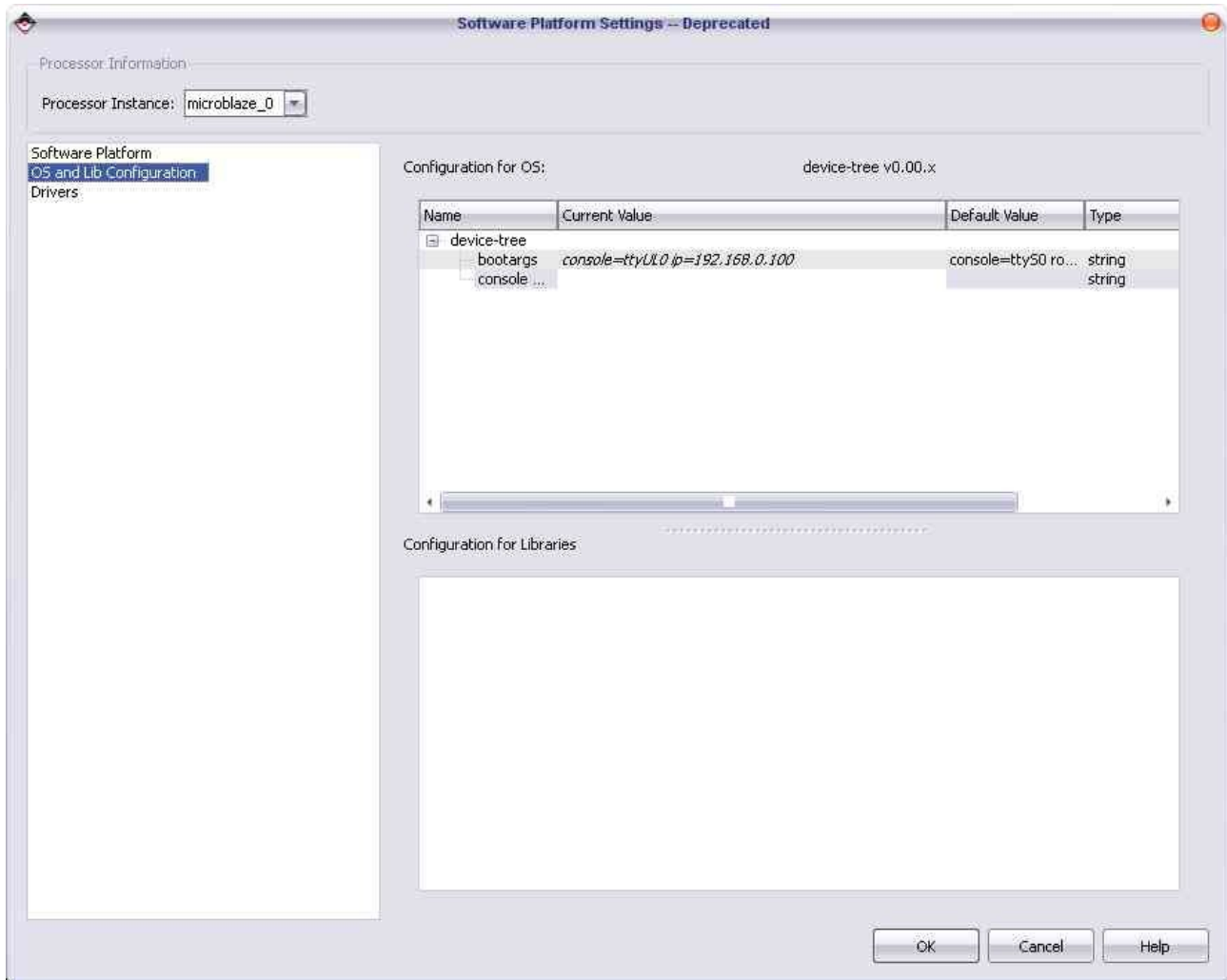


Figure 24 : Configuration du *cmdline* pour la génération du fichier *.dts*

Dans le menu *Software*, il faut enfin sélectionner le choix *Generate Libraries and BSPs* pour générer le fichier *.dts*.

3.3. Installation des outils Xilinx sous Linux

Il est possible d'utiliser les outils Xilinx sous Linux que ce soit pour le design du système SoPC que pour le développement logiciel. Cela permet d'avoir une seule machine sous Linux, ce qui permet de garder une cohérence dans le développement matériel et logiciel.

Il convient de récupérer les outils sur le site de Xilinx (s'inscrire au préalable) [5]. La procédure a été testée sous Fedora 12.

On installera les outils Xilinx ISE version 11.4 (à la date d'écriture de ce document). Il faut installer au préalable la version 11.1 :

```
$ cd
$ tar -xvf Xilinx_11.1_ISE_SFD.tar
$ cd Xilinx_11.1_ISE_SFD
# ./xsetup
$ cd
$ tar -xvf 91_Xilinx_11.1_EDK_SFD.tar
$ cd Xilinx_11.1_EDK_SFD
# ./xsetup
$ cd
$ tar -xvf Xilinx_11.1_SDK_SFD.tar
$ cd Xilinx_11.1_SDK_SFD
# ./xsetup
$ cd
$ tar -xvf Xilinx_11.4_ISE_DS_lin.tar
$ cd Xilinx_11.4_ISE_DS_lin
# ./xsetup
$ cd
```

L'ensemble des logiciels est installé dans le répertoire */opt/Xilinx/11.1*.

On créera le script *mbsdk* que l'on placera sous *~/bin* :

```
#!/bin/bash
export LM_LICENSE_FILE=1700@localhost

export XIL_IMPACT_USE_LIBUSB=1

source /opt/Xilinx/11.1/ISE/settings32.sh
source /opt/Xilinx/11.1/EDK/settings32.sh

# set prompt
export PS1="[MB EDK]$ "

bash
```

Pour pouvoir utiliser les outils Xilinx ISE, il faut bien sûr avoir une licence Xilinx valable que l'on utilisera en réseau avec un serveur *flexlm*...

Mais l'on peut utiliser les autres outils en ligne de commande.

On installera la sonde JTAG :

```
# yum install -y fxload
# cd /opt/Xilinx/11.1/common/bin/lin/
# sh setup_pcusb
```

On vérifiera l'existence du fichier */etc/udev/rules.d/xusbdfwu.rules* pour la règle udev.

La sonde JTAG peut être pilotée avec la bibliothèque *libusb* (on vérifiera que la bibliothèque *libusb* est bien installée, ce qui est le cas par défaut pour Fedora). Il convient de rajouter le lien symbolique suivant :

```
# cd /usr/lib/  
# ln -s libusb-0.1.so.4 libusb.so
```

On vérifie que la commande *xmd* fonctionne pour télécharger un fichier dans la cible par le JTAG :

```
[Xilinx EDK]$ xmd  
Xilinx Microprocessor Debugger (XMD) Engine  
Xilinx EDK 11.4 Build EDK_LS4.68  
Copyright (c) 1995-2009 Xilinx, Inc. All rights reserved.
```

```
XMD%  
XMD% exit
```

Les outils Xilinx sont opérationnels sous Linux.

3.4. Configuration logicielle : configuration et compilation de Linux pour MicroBlaze

Le processeur MicroBlaze étant configuré avec sa MMU, la **distribution Linux standard** est utilisée.

Il convient dans un premier temps de télécharger la distribution Linux pour MicroBlaze. On se référera au document [6] pour plus de détails :

```
$ cd  
$ git clone git://git.xilinx.com/linux-2.6-xlnx.git  
$ ln -s linux-2.6-xlnx mb-linux
```

On suppose que *\$mb_linux* représente le chemin du répertoire *mb-linux*.

On installe le compilateur croisé *gcc* pour le processeur MicroBlaze :

```
$ cd  
$ git clone git://git.xilinx.com/xldk/microblaze_v1.0.git  
$ cd microblaze_v1.0  
$ tar jxpf mb_gnu_tools_bin.tar.bz  
$ su  
# \mv microblaze-unknown-linux-gnu /opt
```

On ajuste sa variable *PATH* dans son fichier *profile* (*~/.bash_profile*) :

```
PATH=$PATH: /opt/microblaze-unknown-linux-gnu/bin  
export PATH
```

On vérifie que le compilateur *gcc* pour MicroBlaze est opérationnel :

```
$ microblaze-unknown-linux-gnu-gcc -v
Using built-in specs.
Target: microblaze-unknown-linux-gnu
Configured with: /home/ddeboni/src/Linux/crosstool/crosstool-ng-1.4.1/targets/src/gcc-4.1.2/configure --build=i386-build_redhat-linux-gnu --host=i386-build_redhat-linux-gnu --target=microblaze-unknown-linux-gnu --prefix=/tmp/microblaze-unknown-linux-gnu --with-sysroot=/tmp/microblaze-unknown-linux-gnu/microblaze-unknown-linux-gnu//sys-root --enable-languages=c,c++ --disable-multilib --enable-__cxa_atexit --with-local-prefix=/tmp/microblaze-unknown-linux-gnu/microblaze-unknown-linux-gnu//sys-root --disable-nls --enable-threads=posix --enable-symvers=gnu --enable-c99 --enable-long-long --enable-target-optspace --enable-multilib
Thread model: posix
gcc version 4.1.2
```

On suppose que *\$design* représente le chemin du répertoire contenant le design XPS. On récupère le fichier de programmation du circuit FPGA *system.bit* et le fichier DTS *xilinx.dts* issus de la synthèse :

```
$ cd $design
$ cp implementation/system.bit $mb_linux/preempt-rt.bit
$ cp microblaze/libsrc/device-tree/xilinx.dts $mb_linux/preempt-rt.dts
```

Le fichier *preempt-rt.dts* est à recopier sous *\$mb_linux/arch/microblaze/boot/dts* :

```
$ cd $mb_linux
$ cp preempt-rt.dts arch/microblaze/boot/dts
```

On utilisera un fichier *initramfs* minimal comme système de fichiers root :

```
$ cd
$ cd microblaze_v1.0
$ cp initramfs_minimal_cpio.gz $mb_linux
```

On configure Linux pour le processeur MicroBlaze :

```
$ cd $mb_linux
$ make ARCH=microblaze CROSS_COMPILE=microblaze-unknown-linux-gnu-xilinx_mmu_defconfig
```

On compile le noyau Linux :

```
$ make ARCH=microblaze CROSS_COMPILE=microblaze-unknown-linux-gnu-simpleImage.preempt-rt
```

On programme le design dans le circuit FPGA par la sonde JTAG :

```
$ mbsdk
[Xilinx EDK]$ cat preempt-rt.cmd
setMode -bscan
setCable -p auto
identify
assignfile -p 3 -file preempt-rt.bit
program -p 3
quit
[Xilinx EDK]$ impact -batch preempt-rt.cmd
```

On télécharge ensuite le fichier *simpleImage.preempt-rt* dans la carte cible par le JTAG :

```
[Xilinx EDK]$ cat preempt-rt.opt
connect mb mdm
dow linux/arch/microblaze/boot/simpleImage.preempt-rt
run
exit
[Xilinx EDK]$ xmd -opt preempt-rt.opt
```

On utilisera *minicom* pour accéder à la liaison série de la carte cible une fois correctement configuré (9600, 8, 1, N) :

L'outil *minicom* étant lancé dans un deuxième terminal, on récupère les traces de boot du noyau Linux :

```
$ minicom
early_printk_console is enabled at 0x84000000
Ramdisk addr 0x00000003, Compiled-in FDT at 0xc02d35f8
Linux version 2.6.31-00593-g4e2d699 (kadionik@linux00) (gcc version 4.1.2) #18 0
setup_cpuinfo: initialising
setup_cpuinfo: No PVR support. Using static CPU info from FDT
setup_memory: max_mapnr: 0x4000
setup_memory: min_low_pfn: 0x50000
setup_memory: max_low_pfn: 0x54000
On node 0 totalpages: 16384
free_area_init_node: node 0, pgdat c0386230, node_mem_map c04a4000
  Normal zone: 128 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 16256 pages, LIFO batch:3
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: console=ttyUL0 ip=192.168.0.100
PID hash table entries: 256 (order: 8, 1024 bytes)
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 60084k/65536k available
NR_IRQS:32
xlnx,xps-intc-1.00.a #0 at 0xc4000000, num_irq=9, edge=0x200
xlnx,xps-timer-1.00.a #0 at 0xc4004000, irq=0
microblaze_timer_set_mode: shutdown
microblaze_timer_set_mode: periodic
Calibrating delay loop... 48.43 BogoMIPS (lpj=242176)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
XGpio: /plb@0/gpio@81440000: registered
XGpio: /plb@0/gpio@81420000: registered
XGpio: /plb@0/gpio@81400000: registered
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 3, 40960 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
NET: Registered protocol family 1
msgmni has been set to 117
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
84000000.serial: ttyUL0 at MMIO 0x84000003 (irq = 3) is a uartlite
console [ttyUL0] enabled
```

Conception détaillée des portages PREEMPT-RT/Microblaze

```
brd: module loaded
xsysace 83600000.sysace: Xilinx SystemACE revision 1.0.12
xsysace 83600000.sysace: capacity: 1000944 sectors
  xsa: xsa1 xsa2
Xilinx SystemACE device driver, major=254
xilinx_emaclite 81000000.ethernet: Device Tree Probing
xilinx_emaclite 81000000.ethernet: MAC address is now 0: a:35:bf:28: 0
xilinx_emaclite 81000000.ethernet: Xilinx EmacLite at 0x81000000 mapped to 0xC42
i2c /dev entries driver
Device Tree Probing 'i2c'
  #0 at 0x81600000 mapped to 0xC40A0000, irq=5
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
  device=eth0, addr=192.168.0.100, mask=255.255.255.0, gw=255.255.255.255,
  host=192.168.0.100, domain=, nis-domain=(none),
  bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
Freeing unused kernel memory: 937k fStarting rcS...
++ Creating device points
++ Mounting filesystem
++ Loading system loggers
++ Starting telnet daemon
rcS Complete
/bin/sh: can't access tty; job control turned off
/ # ps
PID  USER      TIME  COMMAND
  1  0          0:09  init
  2  0          0:00  [kthreadd]
  3  0          0:00  [ksoftirqd/0]
  4  0          0:00  [watchdog/0]
  5  0          0:00  [events/0]
  6  0          0:00  [khelper]
  7  0          0:00  [async/mgr]
  8  0          0:00  [kblockd/0]
  9  0          0:00  [khungtaskd]
 10  0          0:00  [pdflush]
 11  0          0:00  [pdflush]
 12  0          0:00  [kswapd0]
 13  0          0:00  [aio/0]
 14  0          0:00  [nfsiod]
 15  0          0:00  [cifsockd]
 16  0          0:00  [rpciod/0]
 29  0          0:00  /sbin/syslogd
 31  0          0:00  /sbin/klogd
 34  0          0:00  /sbin/telnetd -l /bin/sh
 38  0          0:00  /bin/sh
 39  0          0:00  ps
/ # cat /proc/timer_list
Timer List Version: v0.4
HRTIMER_MAX_CLOCK_BASES: 2
now at 32436930632 nsecs

cpu: 0
clock 0:
  .base:      c0379884
  .index:     0
  .resolution: 10000000 nsecs
  .get_time:  ktime_get_real
```

```
active timers:
clock 1:
  .base:          c03798a8
  .index:         1
  .resolution:    10000000 nsecs
  .get_time:      ktime_get
active timers:

Tick Device: mode:      0
Per CPU device: 0
Clock Event Device: microblaze_clockevent
max_delta_ns:  2147483647
min_delta_ns:  1000
mult:          1677721
shift:         24
mode:          2
next_event:    2147483646999999999 nsecs
set_next_event: microblaze_timer_set_next_event
set_mode:      microblaze_timer_set_mode
event_handler: tick_handle_periodic

/ #
```

3.5. Configuration logicielle : configuration et compilation de PREEMPT-RT pour MicroBlaze

Il convient de resynchroniser les versions du noyau Linux pour MicroBlaze et du patch général PREEMPT-RT. On met déjà à jour le noyau Linux pour MicroBlaze :

```
$ cd $mb_linux
$ git pull
$ git checkout -b v2.6.31 xilinx_v2.6.31
```

On récupère le patch général version 2.6.31.12-rt21 de PREEMPT-RT que l'on applique aux sources du noyau :

```
$ cd ..
$ wget http://www.kernel.org/pub/linux/kernel/projects/rt/patch-2.6.31.12-rt21.bz2
$ cd $mb_linux
$ bzcat ../patch-2.6.31.12-rt21.bz2 | patch -p
```

Les 4 fichiers suivants sont rejetés mais ils seront corrigés par le patch PREEMPT-RT pour MicroBlaze :

```
./Makefile.rej
./kernel/perf_counter.c.rej
./kernel/futex.c.rej
./kernel/time/clockevents.c.rej
```

On récupère le patch PREEMPT-RT 2.6.31.12-rt21-microblaze-1.0-00 pour MicroBlaze que l'on applique aux sources du noyau :

```
$ wget http://www.enseirb.fr/~kadionik/microblaze-preempt-rt/patches/preempt-rt-2.6.31.12-rt21-microblaze-1.0-00.patch
```

```
$ patch -p1 <../preempt-rt-2.6.31.12-rt21-microblaze-1.0-00.patch
```

On recompile enfin le noyau Linux en validant PREEMPT-RT dans le menu du choix du mode de préemption :

```
$ cd $mb_linux
```

```
$ make ARCH=microblaze CROSS_COMPILE=microblaze-unknown-linux-gnu- gconfig
```

On télécharge ensuite le fichier *simpleImage.preempt-rt* dans la carte cible par le JTAG :

```
$ mbsdk
```

```
[Xilinx EDK]$ impact -batch preempt-rt.cmd
```

On utilisera *minicom* pour accéder à la liaison série de la carte cible une fois correctement configuré :

L'outil *minicom* étant lancé dans un deuxième terminal, on récupère les traces de boot du noyau Linux :

```
$ minicom
```

```
Linux version 2.6.31.12-rt21 (kadionik@linux00) (gcc version 4.1.2) #38 PREEMPT0
setup_cpuinfo: initialising
setup_cpuinfo: No PVR support. Using static CPU info from FDT
setup_memory: max_mapnr: 0x4000
setup_memory: min_low_pfn: 0x50000
setup_memory: max_low_pfn: 0x54000
On node 0 totalpages: 16384
free_area_init_node: node 0, pgdat c03956cc, node_mem_map c04b7000
  Normal zone: 128 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 16256 pages, LIFO batch:3
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Kernel command line: console=ttyUL0 ip=192.168.0.100
PID hash table entries: 256 (order: 8, 1024 bytes)
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 60016k/65536k available
Real-Time Preemption Support (C) 2004-2007 Ingo Molnar
Preemptible RCU implementation.
NR_IRQS:32
xlnx,xps-intc-1.00.a #0 at 0xc4000000, num_irq=9, edge=0x200
xlnx,xps-timer-1.00.a #0 at 0xc4004000, irq=0
microblaze_timer_set_mode: shutdown
microblaze_timer_set_mode: periodic
Calibrating delay loop... 48.64 BogoMIPS (lpj=243200)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
XGpio: /plb@0/gpio@81440000: registered
XGpio: /plb@0/gpio@81420000: registered
XGpio: /plb@0/gpio@81400000: registered
microblaze_timer_set_mode: oneshot
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 3, 57344 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
NET: Registered protocol family 1
msgmni has been set to 117
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
```

Conception détaillée des portages PREEMPT-RT/Microblaze

```
io scheduler cfq registered (default)
Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
84000000.serial: ttyUL0 at MMIO 0x84000003 (irq = 3) is a uartlite
console [ttyUL0] enabled
brd: module loaded
xsysace 83600000.sysace: Xilinx SystemACE revision 1.0.12
xsysace 83600000.sysace: capacity: 1000944 sectors
  xsa: xsa1 xsa2
Xilinx SystemACE device driver, major=254
xilinx_emaclite 81000000.ethernet: Device Tree Probing
xilinx_emaclite 81000000.ethernet: MAC address is now 0: a:35:bf:28: 0
xilinx_emaclite 81000000.ethernet: Xilinx EmacLite at 0x81000000 mapped to 0xC42
i2c /dev entries driver
Device Tree Probing 'i2c'
#0 at 0x81600000 mapped to 0xC40A0000, irq=5
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
    device=eth0, addr=192.168.0.100, mask=255.255.255.0, gw=255.255.255.255,
    host=192.168.0.100, domain=, nis-domain=(none),
    bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
Freeing unused kernel memory: 941k freed
/ # ps
PID   USER     TIME   COMMAND
  1  0         0:06   init
  2  0         0:00   [kthreadd]
  3  0         0:00   [sirq-high/0]
  4  0         0:00   [sirq-timer/0]
  5  0         0:00   [sirq-net-tx/0]
  6  0         0:01   [sirq-net-rx/0]
  7  0         0:00   [sirq-block/0]
  8  0         0:00   [sirq-tasklet/0]
  9  0         0:00   [sirq-sched/0]
 10  0         0:00   [sirq-hrtimer/0]
 11  0         0:00   [sirq-rcu/0]
 12  0         0:00   [posixcputmr/0]
 13  0         0:00   [watchdog/0]
 14  0         0:00   [desched/0]
 15  0         0:00   [rcu_sched_grace]
 16  0         0:00   [events/0]
 17  0         0:00   [khelper]
 18  0         0:00   [async/mgr]
 19  0         0:00   [kblockd/0]
 20  0         0:00   [khungtaskd]
 21  0         0:00   [pdflush]
 22  0         0:00   [pdflush]
 23  0         0:00   [kswapd0]
 24  0         0:00   [aio/0]
 25  0         0:00   [nfsiod]
 26  0         0:00   [cifsoplockd]
 27  0         0:00   [irq/4-systemace]
 28  0         0:00   [irq/5-]
 29  0         0:00   [rpciod/0]
 30  0         0:01   [irq/2-eth0]
 31  0         0:00   [irq/3-uartlite]
 46  0         0:00   /sbin/klogd
 49  0         0:00   /sbin/telnetd -l /bin/sh
 53  0         0:00   /bin/sh
```

```
59 0          0:00 ps
/ # cat /proc/timer_list
Timer List Version: v0.4
HRTIMER_MAX_CLOCK_BASES: 2
now at 825509050322 nsecs

cpu: 0
clock 0:
  .base:      c0389920
  .index:     0
  .resolution: 1 nsecs
  .get_time:  ktime_get_real
  .offset:    0 nsecs
active timers:
clock 1:
  .base:      c0389954
  .index:     1
  .resolution: 1 nsecs
  .get_time:  ktime_get
  .offset:    0 nsecs
active timers:
#0: per_cpu_tick_cpu_sched, tick_sched_timer, S:01
# expires at 825510000000-825510000000 nsecs [in 949678 to 949678 nsecs]
#1: def_rt_bandwidth, sched_rt_period_timer, S:01
# expires at 826000000000-826000000000 nsecs [in 490949678 to 490949678 nsecs]
  .expires_next   : 825510000000 nsecs
  .hres_active    : 1
  .nr_events      : 82289
  .nohz_mode      : 0
  .idle_tick      : 0 nsecs
  .tick_stopped   : 0
  .idle_jiffies   : 0
  .idle_calls     : 0
  .idle_sleeps    : 0
  .idle_entrytime : 0 nsecs
  .idle_waketime  : 0 nsecs
  .idle_exittime  : 0 nsecs
  .idle_sleeptime : 0 nsecs
  .last_jiffies   : 0
  .next_jiffies   : 0
  .idle_expires   : 0 nsecs
jiffies: 52550

Tick Device: mode:      1
Per CPU device: 0
Clock Event Device: microblaze_clockevent
max_delta_ns: 2147483647
min_delta_ns: 1000
mult: 1677721
shift: 24
mode: 3
next_event: 825510000000 nsecs
set_next_event: microblaze_timer_set_next_event
set_mode: microblaze_timer_set_mode
event_handler: hrtimer_interrupt

/ #
```

Avec la commande *ps*, on voit bien l'action de PREEMPT-RT, les interruptions étant maintenant considérées comme des threads...

4. MESURES DES PERFORMANCES

Les mesures de performances ont été effectuées sur la carte cible Virtex-4 ML403 de Xilinx.

Il faut rappeler que le processeur a une fréquence de fonctionnement de 100 MHz. Les expériences menées ont consisté en la mise en œuvre du programme de tests *cyclictest* rendu possible avec le support des timers haute résolution *hrtimer* pour l'architecture microblaze dans les sources du noyau Linux.

4.1. Noyau Linux ordinaire

On utilise sur la carte cible le noyau standard version 2.6.31 sans extension PREEMPT-RT.

On utilise le shell script *goperf-ml403* donné en annexe 2 qui permet de tester le noyau suivant le mode opératoire suivant :

Lancement de *cyclictest* en mode fichier de traces avec un intervalle de 10000 μ s (10 ms) à la priorité SCHED_FIFO de 99. Pas de charge CPU pendant les 5 premières secondes, puis lancement du bench *hackbench* pour un groupe de 20 processus pour la charge CPU. A la fin de l'exécution de *hackbench*, pas de charge CPU pendant 5 secondes avant de recommencer 20 fois en tout.

A partir du fichier de traces donnant les temps de latence obtenus avec *cyclictest*, on peut créer 2 types de courbes :

- Courbe histogramme : donne la répartition histogramme des temps de latence (x=valeur du temps de latence, y=nombre de fois obtenu).
- Courbe latence : donne le temps de latence en fonction du numéro d'échantillon (x=numéro d'échantillon, y=latence). Comme on travaille avec un intervalle de 10000 μ s pour *cyclictest*, on a en fait pour l'abscisse accès au temps : temps en seconde = numéro d'échantillon/100.

Pour construire ces 2 types de courbes, on utilise le shell script *gotraite* donné en annexe 3 qui utilise les shell scripts *gohisto*, *golatency* et *make_hist.sh*.

On obtient les courbes suivantes :

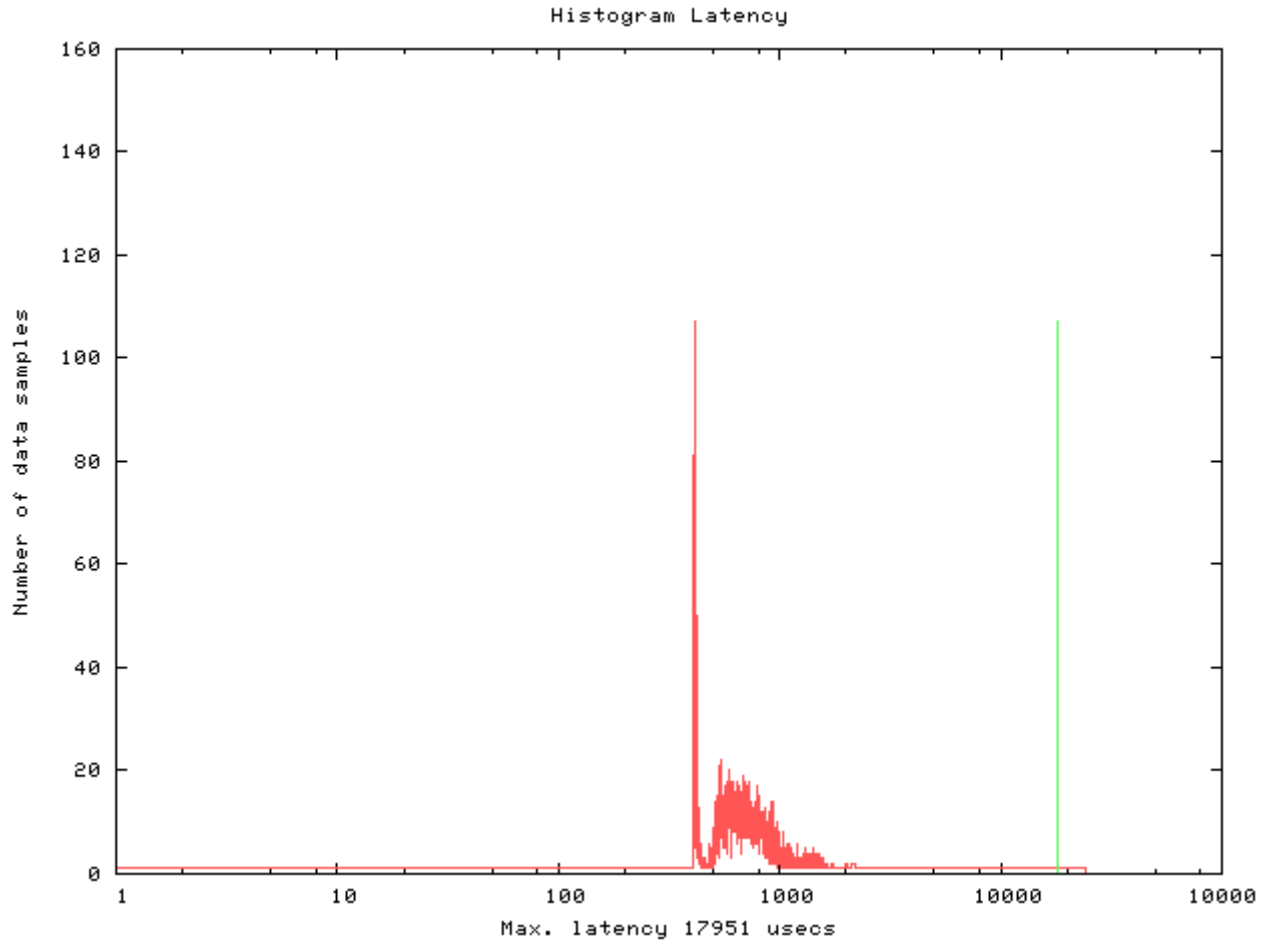


Figure 25 : Courbe histogramme Linux standard

On observe un temps de latence maximal de près de 18 ms.

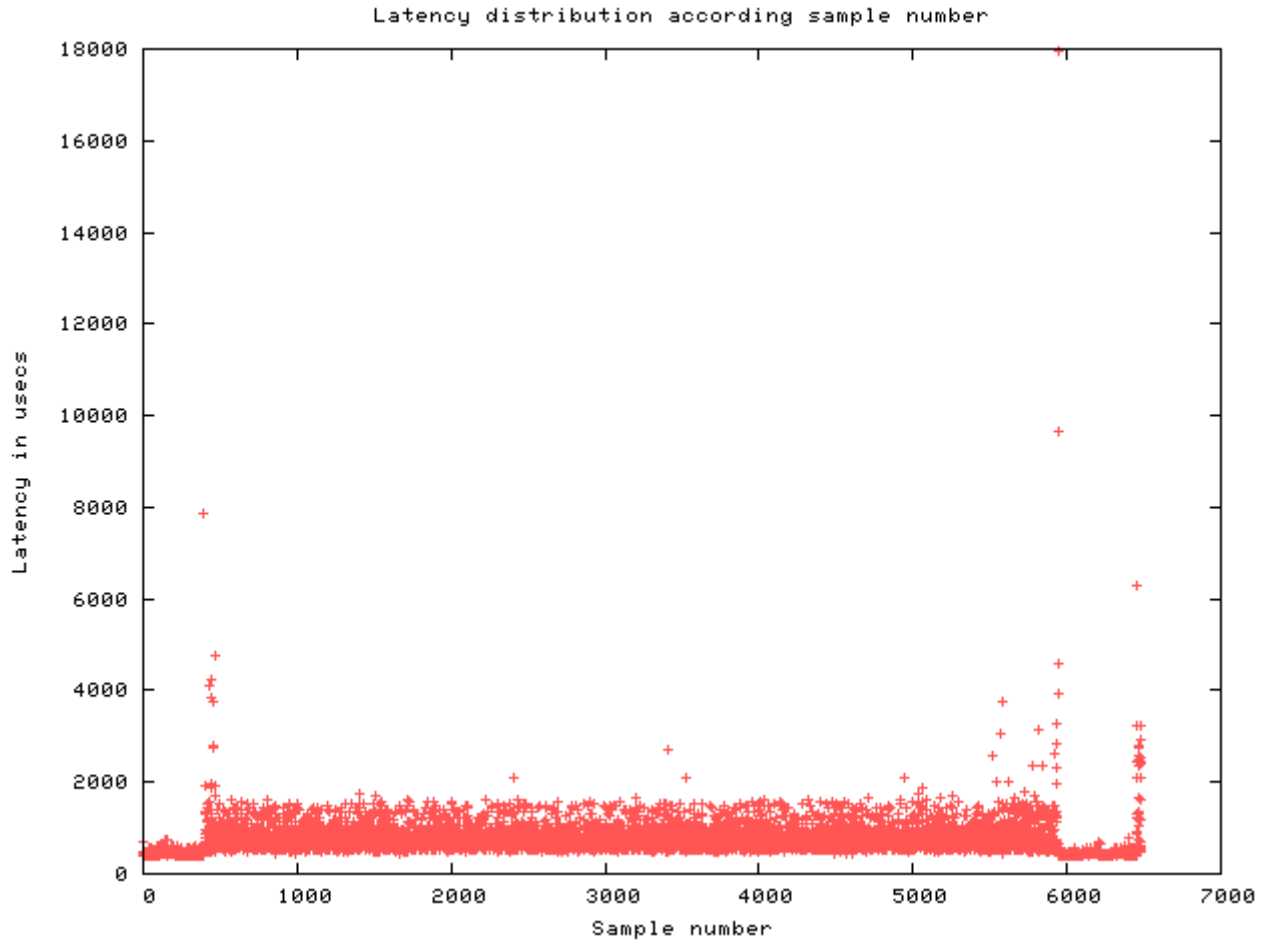


Figure 26 : Courbe latence Linux standard

En divisant par 100 le numéro d'échantillon, on obtient le temps. Entre 0 et 5 secondes, on a un système non chargé, puis de 5 secondes à 60 secondes un système chargé par *hackbench*, puis de 60 secondes à 65 secondes un système non chargé. Le temps de latence est de l'ordre de 400 μ s sans charge pour passer en moyenne à moins de 2 ms. On observe néanmoins des points en dehors de l'épure avec un point à près de 18 ms de temps de latence...

4.2. Noyau Linux PREEMPT-RT

On utilise sur la carte cible le noyau standard version 2.6.31 avec l'extension PREEMPT-RT.

On réalise des mesures avec une valeur de *tick* de 10 ms.

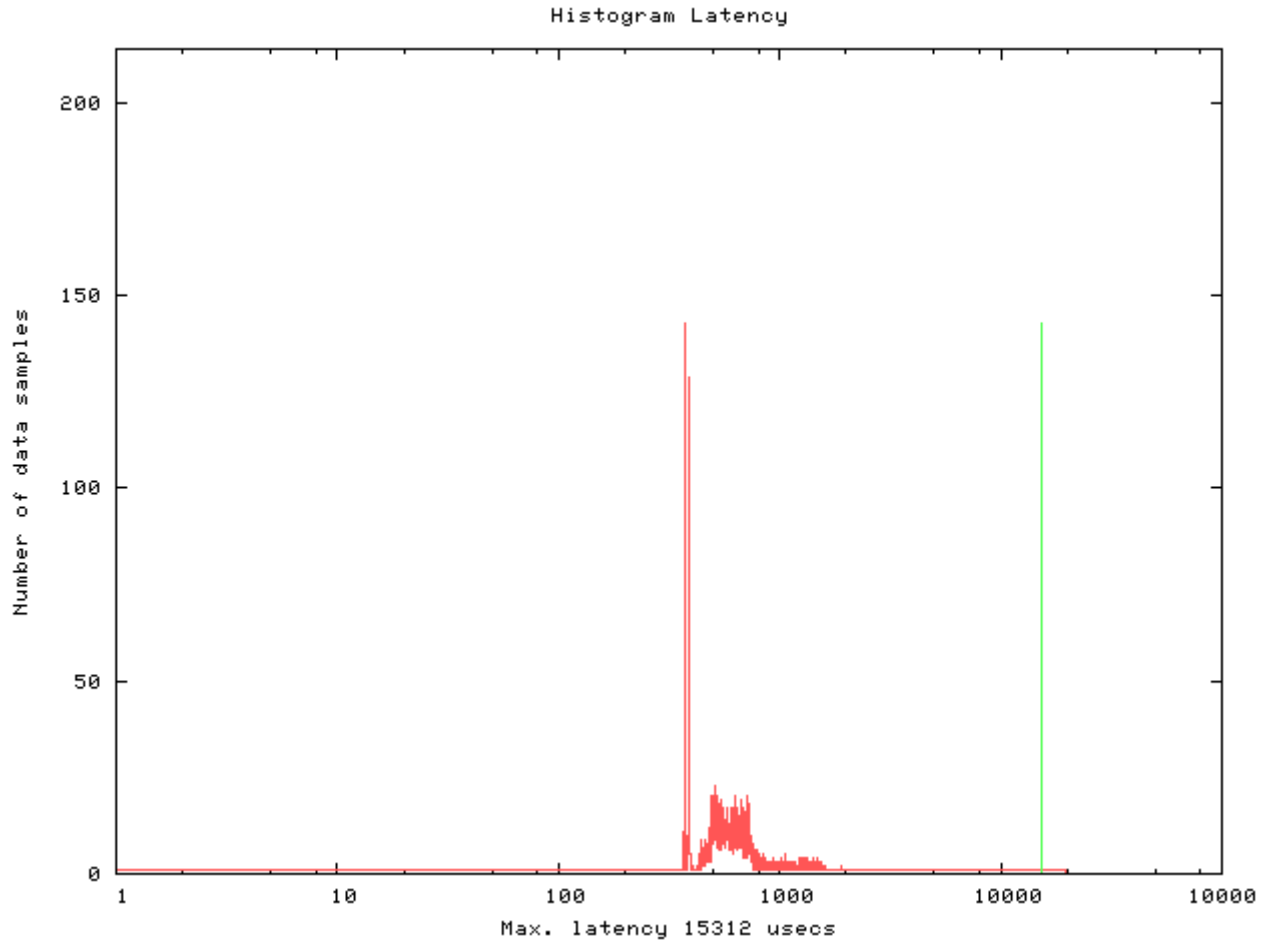


Figure 27 : Courbe histogramme Linux PREEMPT-RT avec *tick* à 10 ms

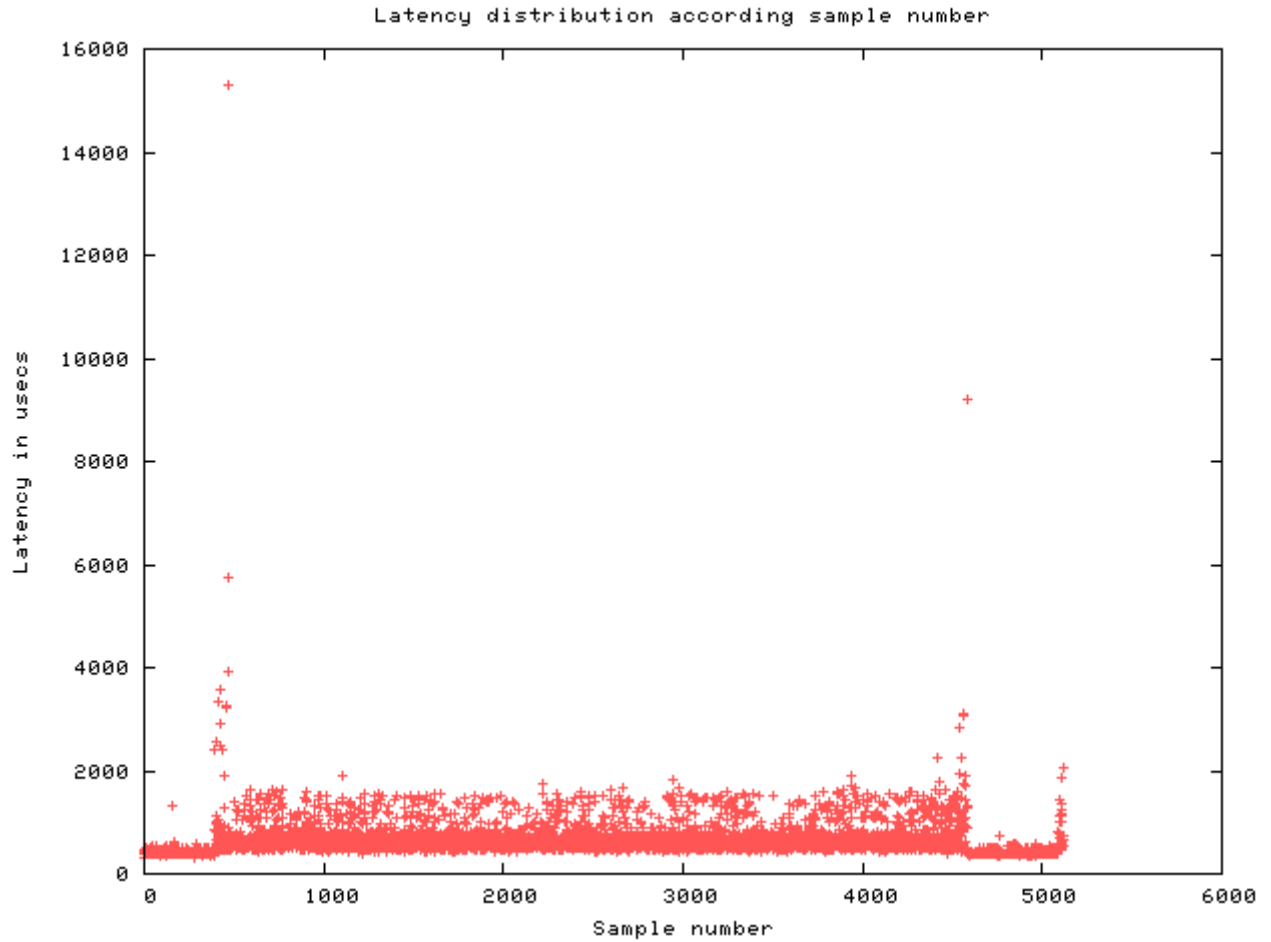


Figure 28 : Courbe latence Linux PREEMPT-RT avec *tick* à 10 ms

Le temps de latence est de l'ordre de 400 μ s sans charge pour passer en moyenne à moins de 2 ms. On observe néanmoins des points en dehors de l'épure avec un point à près de 15 ms de temps de latence...

On recommence les mesures avec un valeur de *tick* de 1 ms.

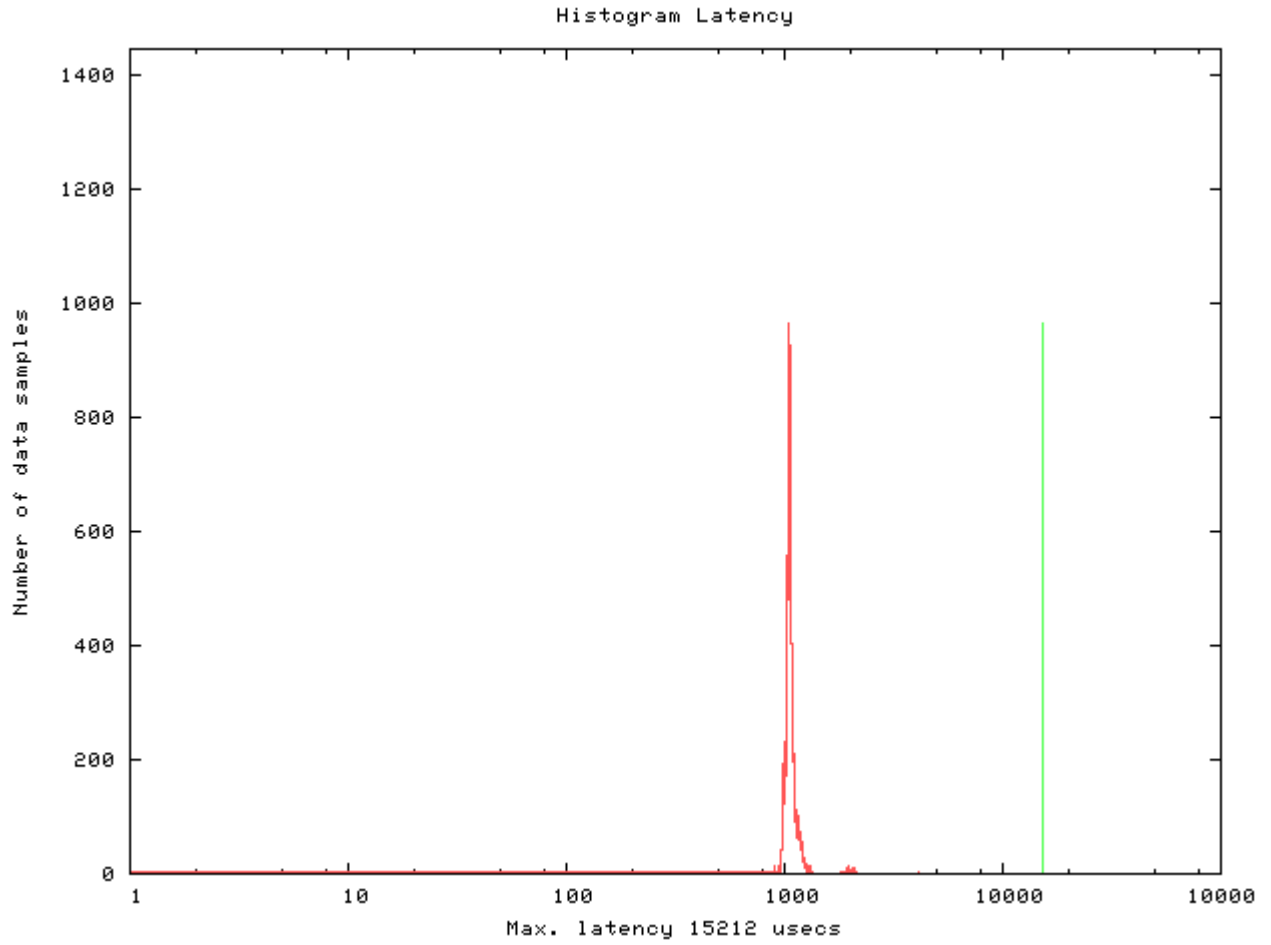


Figure 29 : Courbe histogramme Linux PREEMPT-RT avec *tick* à 1 ms

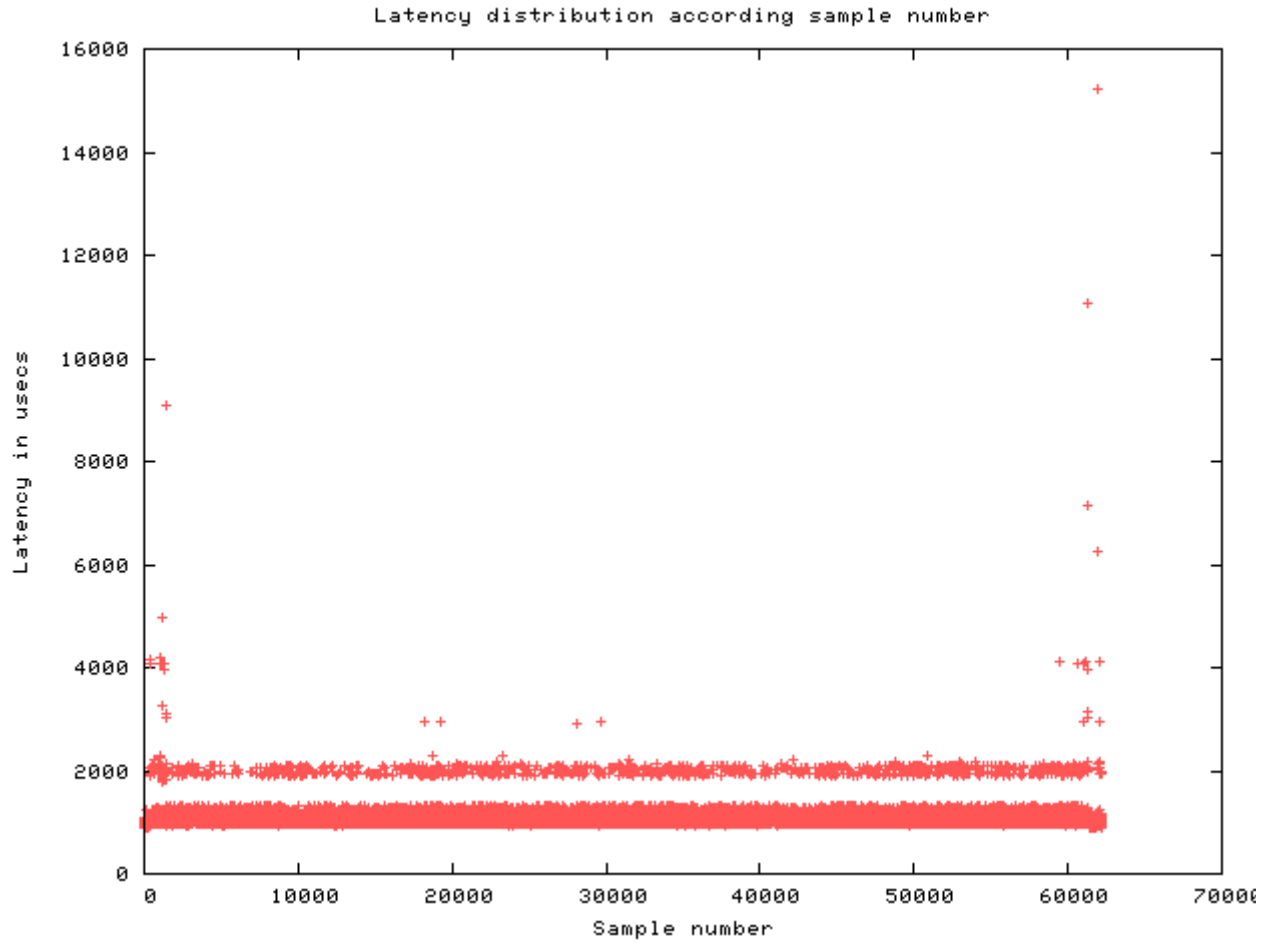


Figure 30 : Courbe latence Linux PREEMPT-RT avec *tick* à 1 ms

On n'observe pas d'amélioration notable du temps de latence...

5. CONCLUSION

Le portage de PREEMPT-RT pour le processeur MicroBlaze dans l'environnement Linux (avec MMU) est opérationnel.

Nous avons aussi défini le design matériel qui a permis le portage de PREEMPT-RT.

Le portage est fonctionnel sur une carte cible de Xilinx (Virtex-4 ML403) fonctionnant à 100 MHz.

L'outil *cyclictest* a été ici utilisé pour les mesures de temps de latence.

L'extension PREEMPT-RT n'améliore pas significativement les temps de latence mesurés avec *cyclictest*.

Une piste possible pour expliquer les résultats est une fréquence CPU très faible ou bien une incompatibilité de l'extension PREEMPT-RT avec l'architecture microblaze.

Pour des environnements Temps Réel dur, il vaudra mieux s'orienter vers la solution Xenomai...

6. RÉFÉRENCES DOCUMENTAIRES

- [1] Projet PREEMPT-RT : https://rt.wiki.kernel.org/index.php/Main_Page
- [2] Projet Linux pour processeur MicroBlaze : <http://xilinx.wikidot.com/microblaze-linux>
- [3] Outils GNU pour processeur MicroBlaze : <http://xilinx.wikidot.com/mb-gnu-tools>
- [4] Page des ressources ENSEIRB-MATMECA de Linux embarqué et du portage de PREEMPT-RT sur processeur MicroBlaze : <http://www.enseirb.fr/~kadionik/microblaze-preempt-rt/>

7. RÉFÉRENCES LOGICIELLES

- [5] Outils Xilinx sous Linux : <http://www.xilinx.com/support/download/index.htm>
- [6] Portage Linux pour le processeur MicroBlaze : <http://xilinx.wikidot.com/microblaze-linux>
- [7] Compilateur croisé gcc pour processeur MicroBlaze : <http://xilinx.wikidot.com/mb-gnu-tools>
- [8] Outil de tests *cyclictest* : <https://rt.wiki.kernel.org/index.php/Cyclictest>
- [9] Outils de bench *hackbench* : <http://devresources.linuxfoundation.org/craiger/hackbench>
- [10] Ces briques logicielles sont en secours sur la page PREEMPT-RT pour le processeur MicroBlaze de l'ENSEIRB-MATMECA : <http://www.enseirb.fr/~kadionik/microblaze-preempt-rt/>

8. ANNEXE 1 : FICHIERS IMPACTÉS PAR LE PORTAGE DE PREEMPT-RT POUR L'ARCHITECTURE MICROBLAZE

8.1. Liste des fichiers génériques modifiés

La liste des fichiers modifiés par le patch PREEMPT-RT concernant les architectures déjà supportées a été supprimée (répertoire *arch/*).

patching file Documentation/kernel-parameters.txt
patching file Documentation/trace/events.txt
patching file Documentation/trace/ftrace.txt
patching file Documentation/trace/function-graph-fold.vim
patching file Documentation/trace/histograms.txt
patching file Documentation/trace/ring-buffer-design.txt
patching file MAINTAINERS
patching file Makefile
patching file arch/Kconfig
patching file block/blk-core.c
patching file drivers/acpi/acpica/acglobal.h
patching file drivers/acpi/acpica/hwregs.c
patching file drivers/acpi/acpica/hwxface.c
patching file drivers/acpi/acpica/utmutex.c
patching file drivers/acpi/ec.c
patching file drivers/acpi/osl.c
patching file drivers/acpi/processor_idle.c
patching file drivers/ata/libata-sff.c
patching file drivers/base/bus.c
patching file drivers/base/core.c
patching file drivers/base/dd.c
patching file drivers/base/power/main.c
patching file drivers/block/hd.c
patching file drivers/block/paride/pseudo.h
patching file drivers/char/random.c
patching file drivers/char/rtc.c
patching file drivers/char/tty_audit.c
patching file drivers/char/tty_buffer.c
patching file drivers/char/tty_ldisc.c
patching file drivers/char/vt.c
patching file drivers/firewire/core-device.c
patching file drivers/gpu/drm/r128/r128_cce.c
patching file drivers/gpu/drm/r128/r128_drv.h
patching file drivers/gpu/drm/r128/r128_state.c
patching file drivers/ide/alim15x3.c
patching file drivers/ide/hpt366.c
patching file drivers/ide/ide-io-std.c
patching file drivers/ide/ide-io.c
patching file drivers/ide/ide-iops.c
patching file drivers/ide/ide-probe.c
patching file drivers/ide/ide-taskfile.c
patching file drivers/ieee1394/nodemgr.c

patching file drivers/infiniband/core/user_mad.c
patching file drivers/infiniband/ulp/ipoib/ipoib_multicast.c
patching file drivers/input/gameport/gameport.c
patching file drivers/input/joystick/analog.c
patching file drivers/input/keyboard/hil_kbd.c
patching file drivers/input/misc/hp_sdc_rtc.c
patching file drivers/input/misc/pcspkr.c
patching file drivers/input/mouse/hil_ptr.c
patching file drivers/input/serio/hil_mlc.c
patching file drivers/input/serio/hp_sdc.c
patching file drivers/macintosh/adb.c
patching file drivers/media/dvb/dvb-core/dvb_frontend.c
patching file drivers/mfd/twl4030-irq.c
patching file drivers/mfd/ucb1x00-core.c
patching file drivers/misc/Kconfig
patching file drivers/misc/Makefile
patching file drivers/misc/hwlat_detector.c
patching file drivers/mmc/card/queue.c
patching file drivers/net/3c527.c
patching file drivers/net/3c59x.c
patching file drivers/net/8139too.c
patching file drivers/net/arm/at91_ether.c
patching file drivers/net/at11c/at11c_main.c
patching file drivers/net/at11e/at11e_main.c
patching file drivers/net/bnx2.c
patching file drivers/net/bnx2x_main.c
patching file drivers/net/chelsio/sge.c
patching file drivers/net/hamradio/6pack.c
patching file drivers/net/hamradio/mkiss.c
patching file drivers/net/irda/sir_dev.c
patching file drivers/net/ixp2000/enp2611.c
patching file drivers/net/ixp2000/ixpdev.c
patching file drivers/net/loopback.c
patching file drivers/net/mlx4/mlx4.h
patching file drivers/net/mv643xx_eth.c
patching file drivers/net/netxen/netxen_nic_init.c
patching file drivers/net/niu.c
patching file drivers/net/ppp_async.c
patching file drivers/net/rionet.c
patching file drivers/net/s2io.c
patching file drivers/net/sungem.c
patching file drivers/net/tehuti.c
patching file drivers/net/tulip/tulip_core.c
patching file drivers/net/wan/cosa.c
patching file drivers/of/base.c
patching file drivers/oprofile/event_buffer.c
patching file drivers/oprofile/oprofilefs.c
patching file drivers/parport/ieee1284.c
patching file drivers/parport/share.c
patching file drivers/pci/access.c
patching file drivers/pci/bus.c
patching file drivers/pci/hotplug/ibmphp_hpc.c

patching file drivers/pci/pci.c
patching file drivers/pcmcia/ds.c
patching file drivers/s390/cio/crw.c
patching file drivers/scsi/aacraid/aacraid.h
patching file drivers/scsi/aacraid/commctrl.c
patching file drivers/scsi/aacraid/commsup.c
patching file drivers/scsi/aacraid/dpcsup.c
patching file drivers/serial/8250.c
patching file drivers/staging/comedi/drivers/dt9812.c
patching file drivers/staging/comedi/drivers/usbdx.c
patching file drivers/staging/comedi/drivers/usbdxfast.c
patching file drivers/staging/cpc-usb/cpc-usb_drv.c
patching file drivers/staging/frontier/alphatrack.c
patching file drivers/staging/frontier/tranzport.c
patching file drivers/staging/go7007/go7007-driver.c
patching file drivers/staging/go7007/go7007-i2c.c
patching file drivers/staging/go7007/go7007-usb.c
patching file drivers/staging/go7007/go7007-v4l2.c
patching file drivers/staging/go7007/s2250-loader.c
patching file drivers/staging/mimio/mimio.c
patching file drivers/staging/octeon/ethernet-mdio.c
patching file drivers/staging/otus/wwrap.c
patching file drivers/staging/p9auth/p9auth.c
patching file drivers/staging/rspiusb/rspiusb.c
patching file drivers/staging/rt2870/common/2870_rtmp_init.c
patching file drivers/usb/core/devio.c
patching file drivers/usb/core/driver.c
patching file drivers/usb/core/hcd.c
patching file drivers/usb/core/message.c
patching file drivers/usb/gadget/inode.c
patching file drivers/usb/misc/ftdi-elan.c
patching file drivers/uwb/umc-bus.c
patching file drivers/uwb/uwb-internal.h
patching file drivers/video/console/fbcon.c
patching file drivers/video/console/vgacon.c
patching file drivers/watchdog/davinci_wdt.c
patching file fs/affs/super.c
patching file fs/aio.c
patching file fs/attr.c
patching file fs/btrfs/locking.c
patching file fs/buffer.c
patching file fs/cifs/cifsglob.h
patching file fs/cifs/cifssmb.c
patching file fs/cifs/connect.c
patching file fs/cifs/misc.c
patching file fs/dcache.c
patching file fs/direct-io.c
patching file fs/exec.c
patching file fs/ext4/inode.c
patching file fs/fat/inode.c
patching file fs/file.c
patching file fs/hfs/bfind.c

patching file fs/hfs/btree.c
patching file fs/hfs/btree.h
patching file fs/hfsplus/bfind.c
patching file fs/hfsplus/btree.c
patching file fs/hfsplus/hfsplus_fs.h
patching file fs/hpfs/buffer.c
patching file fs/hpfs/hpfs_fn.h
patching file fs/hpfs/super.c
patching file fs/inode.c
patching file fs/ioprio.c
patching file fs/jbd/transaction.c
patching file fs/jbd2/transaction.c
patching file fs/namespace.c
patching file fs/nilfs2/mdt.c
patching file fs/ntfs/aops.c
patching file fs/ntfs/file.c
patching file fs/ocfs2/aops.c
patching file fs/ocfs2/file.c
patching file fs/pipe.c
patching file fs/proc/array.c
patching file fs/proc/stat.c
patching file fs/proc/task_mmu.c
patching file fs/reiserfs/xattr.c
patching file fs/smbfs/inode.c
patching file fs/xfs/linux-2.6/mrlock.h
patching file fs/xfs/linux-2.6/xfs_buf.c
patching file fs/xfs/linux-2.6/xfs_buf.h
patching file fs/xfs/xfs_alloc.c
patching file fs/xfs/xfs_bmap.c
patching file fs/xfs/xfs_filestream.c
patching file fs/xfs/xfs_fsops.c
patching file fs/xfs/xfs_ialloc.c
patching file fs/xfs/xfs_itable.c
patching file fs/xfs/xfs_mount.c
patching file fs/xfs/xfs_mount.h
patching file include/acpi/acpiosxf.h
patching file include/asm-generic/bug.h
patching file include/asm-generic/cmpxchg-local.h
patching file include/asm-generic/percpu.h
patching file include/asm-generic/tlb.h
patching file include/asm-generic/vmlinux.lds.h
patching file include/linux/bottom_half.h
patching file include/linux/buffer_head.h
patching file include/linux/console.h
patching file include/linux/device.h
patching file include/linux/fs.h
patching file include/linux/ftrace_event.h
patching file include/linux/hardirq.h
patching file include/linux/hrtimer.h
patching file include/linux/init_task.h
patching file include/linux/interrupt.h
patching file include/linux/irq.h

patching file include/linux/jbd.h
patching file include/linux/jbd2.h
patching file include/linux/kernel.h
patching file include/linux/kernel_stat.h
patching file include/linux/kprobes.h
patching file include/linux/list.h
patching file include/linux/lockdep.h
patching file include/linux/mm.h
patching file include/linux/mm_types.h
patching file include/linux/module.h
patching file include/linux/mutex.h
patching file include/linux/netdevice.h
patching file include/linux/netfilter/x_tables.h
patching file include/linux/netpoll.h
patching file include/linux/oprofile.h
patching file include/linux/page_cgroup.h
patching file include/linux/pagevec.h
patching file include/linux/parport.h
patching file include/linux/percpu-defs.h
patching file include/linux/percpu.h
patching file include/linux/percpu_counter.h
patching file include/linux/perf_counter.h
patching file include/linux/plist.h
patching file include/linux/preempt.h
patching file include/linux/profile.h
patching file include/linux/proportions.h
patching file include/linux/quicklist.h
patching file include/linux/radix-tree.h
patching file include/linux/ring_buffer.h
patching file include/linux/rt_lock.h
patching file include/linux/rtmutex.h
patching file include/linux/rwlock.h
patching file include/linux/rwlock_types.h
patching file include/linux/rwsem-spinlock.h
patching file include/linux/rwsem.h
patching file include/linux/sched.h
patching file include/linux/semaphore.h
patching file include/linux/seqlock.h
patching file include/linux/signal.h
patching file include/linux/skbuff.h
patching file include/linux/smb_fs_sb.h
patching file include/linux/smp.h
patching file include/linux/smp_lock.h
patching file include/linux/spinlock.h
patching file include/linux/spinlock_api_smp.h
patching file include/linux/spinlock_api_up.h
patching file include/linux/spinlock_types.h
patching file include/linux/srcu.h
patching file include/linux/syscalls.h
patching file include/linux/time.h
patching file include/linux/timer.h
patching file include/linux/timex.h

patching file include/linux/topology.h
patching file include/linux/tracepoint.h
patching file include/linux/tty.h
patching file include/linux/uaccess.h
patching file include/linux/usb.h
patching file include/linux/vmstat.h
patching file include/linux/workqueue.h
patching file include/net/bluetooth/hci_core.h
patching file include/trace/define_trace.h
patching file include/trace/events/block.h
patching file include/trace/events/hist.h
patching file include/trace/events/latency_hist.h
patching file include/trace/events/module.h
patching file include/trace/events/sched.h
patching file include/trace/events/syscalls.h
patching file include/trace/ftrace.h
patching file include/trace/syscall.h
patching file init/Kconfig
patching file init/Makefile
patching file init/main.c
patching file ipc/mqueue.c
patching file ipc/msg.c
patching file ipc/sem.c
patching file kernel/Kconfig.preempt
patching file kernel/Makefile
patching file kernel/audit.c
patching file kernel/capability.c
patching file kernel/cgroup.c
patching file kernel/exit.c
patching file kernel/fork.c
patching file kernel/futex.c
patching file kernel/futex_compat.c
patching file kernel/hrtimer.c
patching file kernel/irq/autoprobe.c
patching file kernel/irq/chip.c
patching file kernel/irq/handle.c
patching file kernel/irq/internals.h
patching file kernel/irq/manage.c
patching file kernel/irq/migration.c
patching file kernel/irq/numa_migrate.c
patching file kernel/irq/pm.c
patching file kernel/irq/proc.c
patching file kernel/irq/spurious.c
patching file kernel/itimer.c
patching file kernel/kmod.c
patching file kernel/kprobes.c
patching file kernel/latencytop.c
patching file kernel/lock-internals.h
patching file kernel/lockdep.c
patching file kernel/module.c
patching file kernel/mutex-debug.h
patching file kernel/mutex.c

patching file kernel/notifier.c
patching file kernel/perf_counter.c
patching file kernel/posix-cpu-timers.c
patching file kernel/posix-timers.c
patching file kernel/printk.c
patching file kernel/rcupreempt.c
patching file kernel/rcutorture.c
patching file kernel/relay.c
patching file kernel/res_counter.c
patching file kernel/rt.c
patching file kernel/rtmutex-debug.c
patching file kernel/rtmutex-debug.h
patching file kernel/rtmutex.c
patching file kernel/rtmutex_common.h
patching file kernel/rwlock.c
patching file kernel/rwsem.c
patching file kernel/sched.c
patching file kernel/sched_cpupri.c
patching file kernel/sched_cpupri.h
patching file kernel/sched_debug.c
patching file kernel/sched_fair.c
patching file kernel/sched_idletask.c
patching file kernel/sched_rt.c
patching file kernel/sched_stats.h
patching file kernel/semaphore.c
patching file kernel/signal.c
patching file kernel/smp.c
patching file kernel/softirq.c
patching file kernel/softlockup.c
patching file kernel/spinlock.c
patching file kernel/srcu.c
patching file kernel/stop_machine.c
patching file kernel/sys.c
patching file kernel/sysctl.c
patching file kernel/time.c
patching file kernel/time/clockevents.c
patching file kernel/time/clocksource.c
patching file kernel/time/ntp.c
patching file kernel/time/tick-broadcast.c
patching file kernel/time/tick-common.c
patching file kernel/time/tick-internal.h
patching file kernel/time/tick-sched.c
patching file kernel/time/timekeeping.c
patching file kernel/time/timer_list.c
patching file kernel/time/timer_stats.c
patching file kernel/timer.c
patching file kernel/trace/Kconfig
patching file kernel/trace/Makefile
patching file kernel/trace/blktrace.c
patching file kernel/trace/ftrace.c
patching file kernel/trace/kmemtrace.c
patching file kernel/trace/latency_hist.c

patching file kernel/trace/ring_buffer.c
patching file kernel/trace/trace.c
patching file kernel/trace/trace.h
patching file kernel/trace/trace_boot.c
patching file kernel/trace/trace_entries.h
patching file kernel/trace/trace_event_profile.c
patching file kernel/trace/trace_event_types.h
patching file kernel/trace/trace_events.c
patching file kernel/trace/trace_events_filter.c
patching file kernel/trace/trace_export.c
patching file kernel/trace/trace_functions.c
patching file kernel/trace/trace_functions_graph.c
patching file kernel/trace/trace_irqsoff.c
patching file kernel/trace/trace_mmio.c
patching file kernel/trace/trace_output.c
patching file kernel/trace/trace_output.h
patching file kernel/trace/trace_power.c
patching file kernel/trace/trace_sched_switch.c
patching file kernel/trace/trace_sched_wakeup.c
patching file kernel/trace/trace_selftest.c
patching file kernel/trace/trace_stack.c
patching file kernel/trace/trace_stat.c
patching file kernel/trace/trace_stat.h
patching file kernel/trace/trace_syscalls.c
patching file kernel/trace/trace_workqueue.c
patching file kernel/tracepoint.c
patching file kernel/user.c
patching file kernel/workqueue.c
patching file lib/Kconfig
patching file lib/Kconfig.debug
patching file lib/Makefile
patching file lib/debugobjects.c
patching file lib/dec_and_lock.c
patching file lib/kernel_lock.c
patching file lib/locking-selftest.c
patching file lib/percpu_counter.c
patching file lib/plist.c
patching file lib/proportions.c
patching file lib/radix-tree.c
patching file lib/ratelimit.c
patching file lib/rwsem-spinlock.c
patching file lib/rwsem.c
patching file lib/scatterlist.c
patching file lib/spinlock_debug.c
patching file mm/bounce.c
patching file mm/filemap.c
patching file mm/highmem.c
patching file mm/memcontrol.c
patching file mm/memory.c
patching file mm/mmap.c
patching file mm/oom_kill.c
patching file mm/page_alloc.c

patching file mm/page_cgroup.c
patching file mm/quicklist.c
patching file mm/slab.c
patching file mm/swap.c
patching file mm/vmscan.c
patching file mm/vmstat.c
patching file net/bluetooth/hci_core.c
patching file net/core/dev.c
patching file net/core/flow.c
patching file net/core/netpoll.c
patching file net/core/skbuff.c
patching file net/core/sock.c
patching file net/ipv4/icmp.c
patching file net/ipv4/netfilter/arp_tables.c
patching file net/ipv4/netfilter/ip_tables.c
patching file net/ipv4/route.c
patching file net/ipv4/tcp.c
patching file net/ipv6/netfilter/ip6_tables.c
patching file net/netfilter/core.c
patching file net/netlink/af_netlink.c
patching file net/sched/sch_generic.c
patching file scripts/Kbuild.include
patching file scripts/checkpatch.pl
patching file scripts/mkcompile_h
patching file scripts/recordmcount.pl
patching file security/keys/permission.c
patching file security/keys/proc.c
patching file sound/drivers/pcsp/pcsp.h
patching file sound/drivers/pcsp/pcsp_input.c
patching file sound/drivers/pcsp/pcsp_lib.c
patching file sound/soc/s3c24xx/s3c2443-ac97.c
patching file tools/perf/util/parse-events.c
patching file virt/kvm/kvm_main.c

8.2. Liste des fichiers spécifiques modifiés pour l'architecture microblaze

patching file arch/microblaze/include/asm/prom.h
patching file arch/microblaze/kernel/entry.S
patching file arch/microblaze/kernel/exceptions.c
patching file arch/microblaze/kernel/process.c
patching file arch/microblaze/kernel/prom.c
patching file arch/microblaze/kernel/signal.c
patching file arch/microblaze/mm/fault.c
patching file drivers/gpio/gpiolib.c
patching file drivers/gpio/xilinx_gpio.c
patching file drivers/i2c/algos/xilinx_iic/i2c-algo-xilinx.c
patching file drivers/i2c/i2c-dev.c
patching file drivers/net/xilinx_emaclite.c
patching file drivers/serial/uartlite.c
patching file drivers/spi/spi_bitbang.c

patching file include/asm-generic/bitops/atomic.h
patching file kernel/futex.c
patching file kernel/perf_counter.c
patching file kernel/time/clockevents.c
patching file Makefile

9. ANNEXE 2 : SHELL SCRIPT GOPERF-ML403

```
#!/bin/sh

i=1
hb=1

while [ $i -le 20 ]
do
    echo "Run #${i}..."
    ./cyclicttest -n -p 99 -i 10000 -v > $i.log &
    sleep 5

    echo "hackbench $hb..."
    ./hackbench $hb

    sleep 5
    killall cyclicttest

    i=`expr $i + 1`
done
```

Usage :

Lancement de *cyclicttest* en mode fichier de traces avec un intervalle de 10000 μ s (10 ms) à la priorité SCHED_FIFO de 99. Pas de charge CPU pendant les 5 premières secondes, puis lancement du bench *hackbench* pour un groupe de 20 processus pour la charge CPU. A la fin de l'exécution de *hackbench*, pas de charge CPU pendant 5 secondes avant de recommencer 20 fois en tout :

```
$ ./goperf-ml403
```

10. ANNEXE 3 : SHELL SCRIPT GOTRAITE

```
#!/bin/sh

for i in brut/*.log; do

    echo $i
    \cp $i cyclicdata.log
    ./golatency
    cp latency.pbm brut/lat-`basename $i log`.pbm

    ./gohist
    cp histo.pbm brut/histo-`basename $i log`.pbm
done
```

11. ANNEXE 4 : SHELL SCRIPT GOLATENCY

```
#!/bin/bash

defaultcyclicdata=cyclicdata

cyclicdata=$1
if test -z "$cyclicdata"
then
    cyclicdata=$defaultcyclicdata
fi

plot=latency.pbm

echo "Creating latency plots (may take a while)..."
#./make_hist.sh -i ${cyclicdata}.log 1>/dev/null 2>/dev/null
./make_hist.sh -i ${cyclicdata}.log

latency=${cyclicdata}_plot_thread0.log

echo -e "set title \"Latency distribution according sample
number\"\n\
set terminal pbm color\n\
set xlabel \"Sample number\"\n\
#set logscale x\n\
#set logscale y\n\
set xrange [1:*\n\
set yrange [1:*\n\
set ylabel \"Latency in usecs\"\n\
set output \"\$plot\"\n\
plot \"\$latency\" notitle " | gnuplot -persist

#display $plot
#gimp $plot
```

12. ANNEXE 5 : SHELL SCRIPT GOHISTO

```
#!/bin/bash

defaultcyclicdata=cyclicdata.log

cyclicdata=$1
if test -z "$cyclicdata"
then
    cyclicdata=$defaultcyclicdata
fi

latencydata=latencydata
maxlatencydata=maxlatencydata
plot=histo.pbm

#echo $cyclicdata
echo "Creating histo plots (may take a while)..."

cat $cyclicdata | cut -d: -f3 | sort -n | uniq -c | sed 's/^[ ]
*/g' > latencydata
MAXIMUM_COUNT=`cat $latencydata | awk '{print $1}' | sort -n -u |
tail -1`
MAXIMUM=`tail -1 $latencydata | awk '{print $2}'`
echo MAXIMUM=$MAXIMUM

if [ -z "$(head -1 $latencydata | awk '{print $2}')" ]
then
    sed --in-place 'ld' $latencydata
fi

let YRANGE=$MAXIMUM_COUNT+$MAXIMUM_COUNT/2

echo -e $YRANGE\\t$MAXIMUM >$maxlatencydata

echo -e "set title \"Histogram Latency\"\n\
set terminal pbm color\n\
set xlabel \"Max. latency $MAXIMUM usecs\"\n\
set logscale x\n\
set xrange [1:*\n\
set yrange [0:$YRANGE]\n\
set ylabel \"Number of data samples\"\n\
set output \"$plot\"\n\
plot \"$latencydata\" using 2:1 notitle with
histeps, \"$maxlatencydata\" using 2:1 notitle with boxes" |
gnuplot -persist
```

13. ANNEXE 6 : SHELL SCRIPT MAKE_HIST.SH

```
#!/bin/bash

#
# Copyright (C) 2006,2007 Luotao Fu, Pengutronix
# (lfu@pengutronix.de)
#
# parse the output file of cyclicttest in debug mode in plottable
# histogramm
# files, single files are automatically created on detecting new
# threads.
#
# This program is free software; you can redistribute it and/or
# modify
# it under the terms of the GNU General Public License version 2
# as
# published by the Free Software Foundation;
#
# Software distributed under the License is distributed on an "AS
# IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or
# implied. See the License for the specific language governing
# rights and limitations under the License.

set -e

args=$(getopt i: $*)

if [ $? -ne 0 ] || [ $# -eq 0 ];then
    echo "Usage: $(basename $0) -i inputfile"
    exit 1
fi
for i in $args; do
    case "$i" in
        -i) shift;logfile_name=$(basename $1);;
    esac
done

logfile_name_prefix=$(echo $logfile_name | cut -d . -f 1)

step_dist=2
thread_sum=0
step_counter=0

IFS_BUF=$IFS
IFS="$IFS:"

echo -n "splitting single thread logs"
```

```
while read thread_nr loop_count m_result ;do
    echo "${loop_count} ${m_result}" >> $
{logfile_name_prefix}_plot_thread${thread_nr}".log"
    #determine thread summary and search for min, max
    if [ ${loop_count} -eq 0 ];then
#       if [ ! $thread_nr ] || [ ! $loop_count ] || [ !
$m_result ];then
#           echo "parsing failed, check your input file"
#           exit 1
#       fi
        (( thread_sum++ ))
        if [ -e ${logfile_name_prefix}_plot_thread${thread_nr}".log" ]
; then
            rm ${logfile_name_prefix}_plot_thread${thread_nr}".log"
        fi
    elif [ $loop_count -eq 1 ];then
        max[thread_nr]=${m_result}
        min[thread_nr]=${m_result}
    else
        if [ ${m_result} -gt ${max[thread_nr]} ];then
            max[thread_nr]=${m_result}
        elif [ ${m_result} -lt ${min[thread_nr]} ];then
            min[thread_nr]=${m_result}
        fi
    fi

#now try to collect histogram data
if [ $loop_count -ne 0 ];then
    hist_index=$(( ${m_result}/${step_dist} ))
    tmp_name=hist_data_${thread_nr}[$hist_index]
    tmp_val=${!tmp_name}
    (( tmp_val++ ))
    eval ${tmp_name}=${tmp_val}
fi

#we'd give the user some lifesign every 5000 lines
if [ $(( ${loop_count} / 5000 )) -ge $step_counter ]; then
    echo -n "."
    (( step_counter++ ))
fi
done < $1
echo done

# PK
# Histogramme fait avec autre script
#
exit

echo -n "making histogram files"
```

```
for ((thr_co=0; thr_co < thread_sum - 1; thr_co++)); do
  echo -n "."
  hist_array=hist_data_${thr_co}[@]
  hist_index=0;
  if [ -e ${logfile_name_prefix}_hist_thread${thr_co}.log ];
then
  rm ${logfile_name_prefix}_hist_thread${thr_co}.log"
  fi
  #hist_index_max=$(( ${max[$thr_co]}/${step_dist} ))
  hist_index_max=$(( ${max[$thr_co]}/${step_dist} ))

  for ((i=0; i < ${hist_index_max} ; i++)); do
    var_pnt=hist_data_${thr_co}[$i]
    index_value=$(( ${i} * ${step_dist} + ${min[$thr_co]} ))
    if [ ! -z ${!var_pnt} ]; then
      echo "$index_value ${!var_pnt}" >> $
{logfile_name_prefix}_hist_thread${thr_co}.log"
    fi
  done
done

echo done

IFS=$IFS_BUF
```