Institut Polytechnique de Bordeaux

- PROJET RTEL4I -

GUIDE D'UTILISATION DE XENOMAI SUR PROCESSEUR SOFTCORE NIOS II

Auteur	**:**	P. Kadionik
Version	**:**	1.0
Date	**:**	15/02/11

Laboratoire IMS – IPB ENSEIRB-MATMECA – Université de Bordeaux 1
Domaine Universitaire 33405 Talence

SOMMAIRE

1. Contexte du projet RTEL4I 4

2. Présentation du processeur softcore NIOS II d'Altera 5

3. Mise en œuvre de Xenomai sur le processeur softcore NIOS II 9

3.1. Carte cible 9

3.2. Configuration matérielle : construction du système SoPC 10

- Design de référence 10
- Mapping mémoire 14
- Configuration du processeur NIOS II 14
- Timer hrtimer 20
- Timer hrclock 21
- Timer sys_clk_timer 22

3.3. Installation des outils Altera sous Linux 23

3.4. Configuration logicielle : configuration et compilation de Linux pour NIOS II 25

3.5. Configuration logicielle : configuration et compilation de Xenomai pour NIOS II 28

4. Conclusion 32

5. Références documentaires 33

6. Références logicielles : 33

Fiche synthétique

Matériel	
Processeur cible	NIOS II
Fondeur	Altera
Outils de synthèse	Quartus II version 9.1 et supérieur
Carte cible	Altera Stratix 1S10
Logiciel	
Distribution Linux	Fedora 14
Version Linux pour NIOS II	µClinux 2.6.30
Version compilateur croisé pour NIOS II	3.4.6
Version patch *ipipe* pour NIOS II	adeos-ipipe-2.6.30-nios2-1.1-00.patch
Version Xenomai	2.5.2

Gestion du document :

Date	**Version**	**Modifications**	**Relecture**	**Commentaires**
15/02/11	1.0	PK	PK	Création fichier

1. CONTEXTE DU PROJET RTEL4I

Le projet RTEL4I (Real *Time Embedded Linux for Industries*) se propose de créer un environnement intégré (*framework*) permettant de simplifier la migration d'applications embarquées et Temps Réel depuis un système d'exploitation embarqué Temps Réel propriétaire (VxWorks, pSOS) vers un environnement Linux. Le même environnement pourra bien évidemment être utilisé pour le développement d'applications natives conçues directement pour Linux.

Les partenaires du projet sont MANDRIVA, OPEN WIDE (chef du projet) et SAGEM COMMUNICATIONS.

Le projet RTEL4I est financé par le Pôle SYSTEM@TIC PARIS-REGION dans le Groupe Thématique Logiciel Libre.

Par sa forte expertise dans le domaine des systèmes embarqués et des Logiciels Libres, l'ENSEIRB-MATMECA de l'Institut Polytechnique de Bordeaux, via son laboratoire de recherche IMS, est partenaire du projet RTEL4I.

2. PRESENTATION DU PROCESSEUR SOFTCORE NIOS II D'ALTERA

Le processeur NIOS II (2ème génération du processeur NIOS) est un processeur RISC *softcore* entièrement synchrone, son architecture interne étant de type Harvard. Il possède au maximum 6 niveaux de pipeline, cadencé à quelques dizaines de MHz, avec une largeur de bus de 32 bits. Ses performances sont de 30 à 80 MIPS (*Million Instructions per Second*).

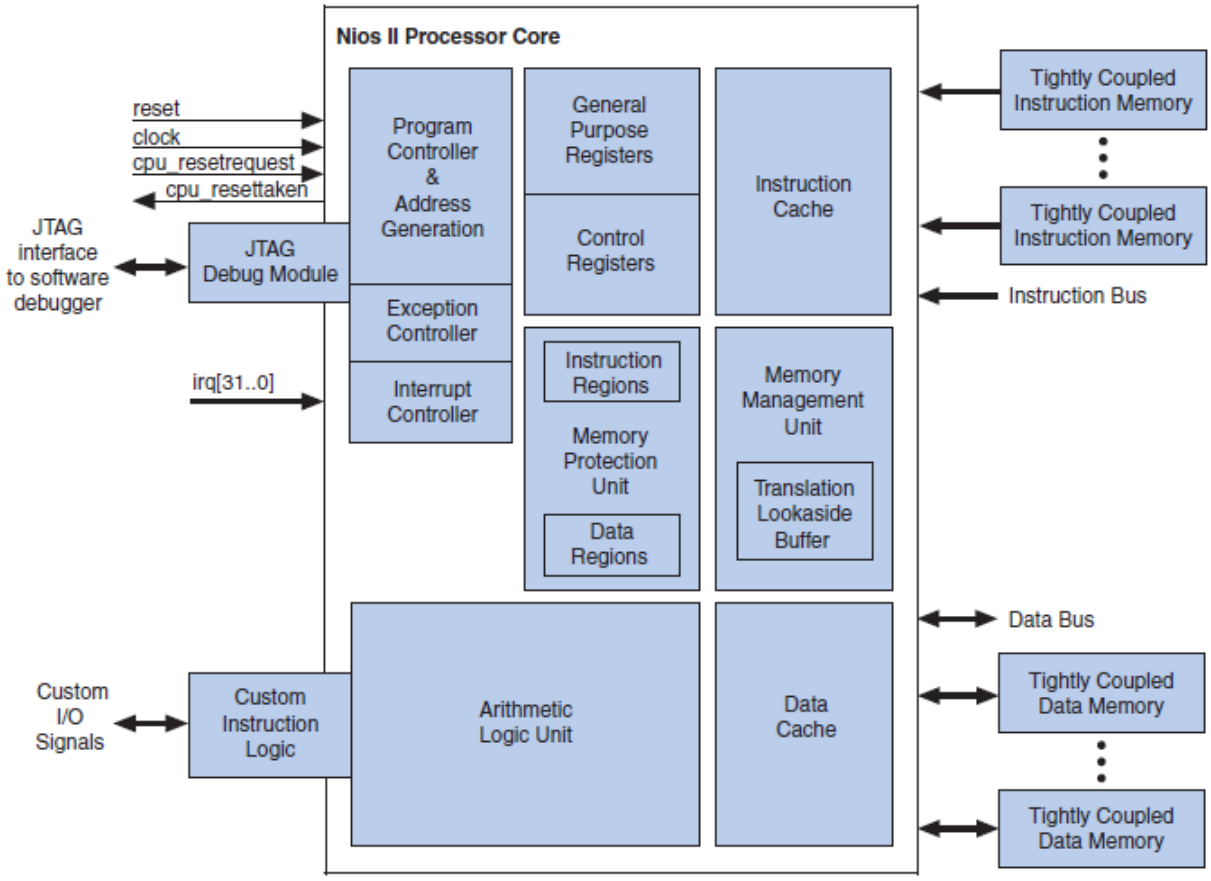


Figure 1 : Architecture du processeur NIOS II

Il est possible d'accélérer certains traitements en ajoutant des instructions personnelles ou C*ustom Instructions* (décrites en VHDL) au processeur NIOS II. De cette manière, il est possible de réaliser la surcharge d'opérateurs ou simplement d'étendre le jeu d'instructions.

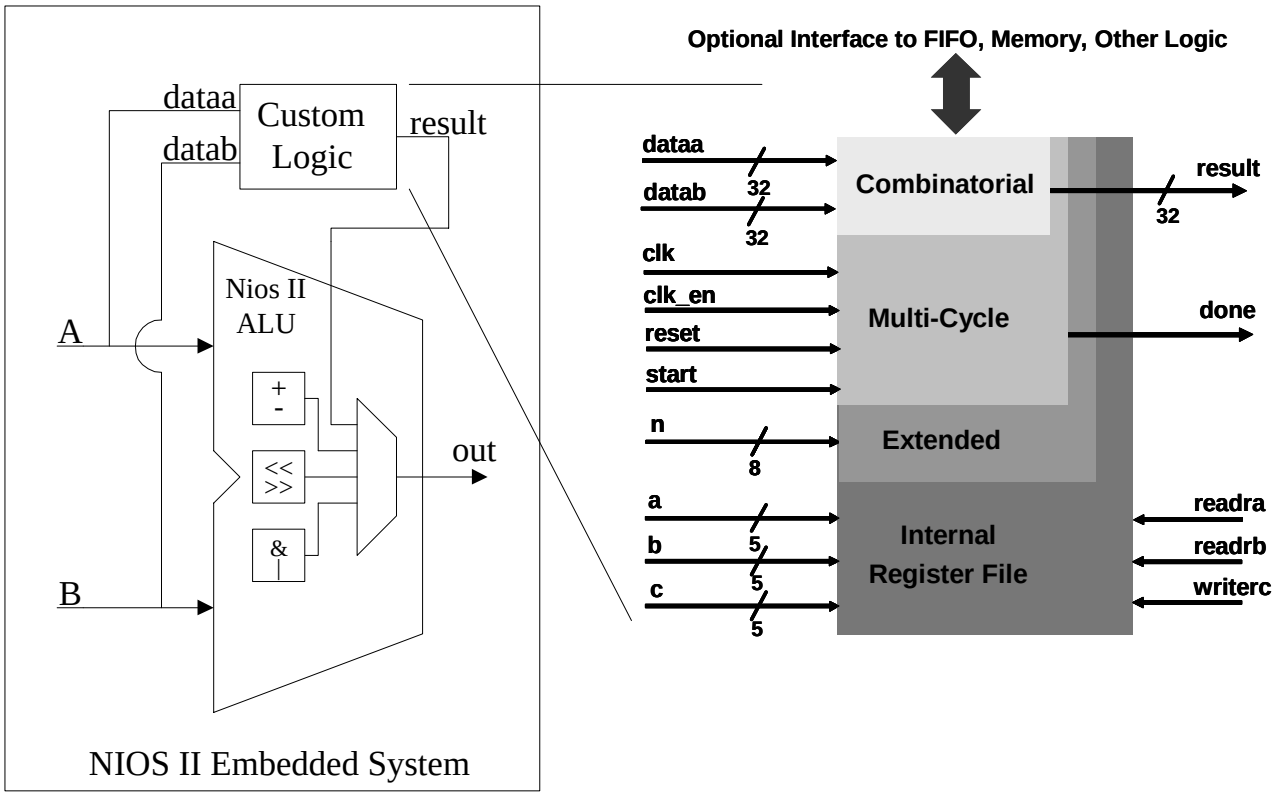


Figure 2 : Instruction personnalisée avec le processeur NIOS II

La table suivante résume les caractéristiques du processeur NIOS II.

NIOS (obsolète)	NIOS II
■ *Architecture RISC pipeline* ■ *Instructions 16 bits* ■ *Liste de registres avec fenêtrage* ■ *Données sur 16 ou 32 bits* ■ *64 niveaux d'interruption* ■ *Cache d'instructions et de données* ■ *Instructions personnalisées*	■ Architecture RISC pipeline ■ Instructions 32 bits ■ Liste de registres fixe ■ Données sur 32 bits ■ 32 niveaux d'interruption ■ Cache d'instructions et de données ■ Instructions personnalisées

Figure 3 : Caractéristiques du processeur NIOS II

Lors de la configuration du processeur NIOS II avec l'outil *SoPC Builder*, il est possible de choisir entre trois versions du processeur NIOS II : une première version *Economy* qui utilise moins de surface de silicium du composant FPGA, une deuxième version *Standard* qui permet un compromis

entre surface et rapidité et enfin une dernière version *Fast* qui est la plus rapide des deux autres versons.

	NIOS II /f	**NIOS II /s**	**NIOS II /e**
Pipeline	6 niveaux	5 niveaux	Non
Multiplication Matériel	1 Cycle	3 Cycle	Par logiciel
Branch Prediction	Dynamic	Static	Non
Cache d'Instructions	Configurable	Configurable	Non
Cache de données	Configurable	Non	Non
Instructions Personnalisés	Supérieur à 256		

Figure 4 : Les trois versions du processeur NIOS II

La figure suivante présente les performances en DMIPS *(Dhrystons Million Instructions per Second*) et la surface occupée en éléments logiques Altera (*Logic Element*) des différentes versions du processeur NIOS II sur différentes familles de composants FPGA d'Altera (Stratix, Stratix II, Cyclone, Cyclone II…).

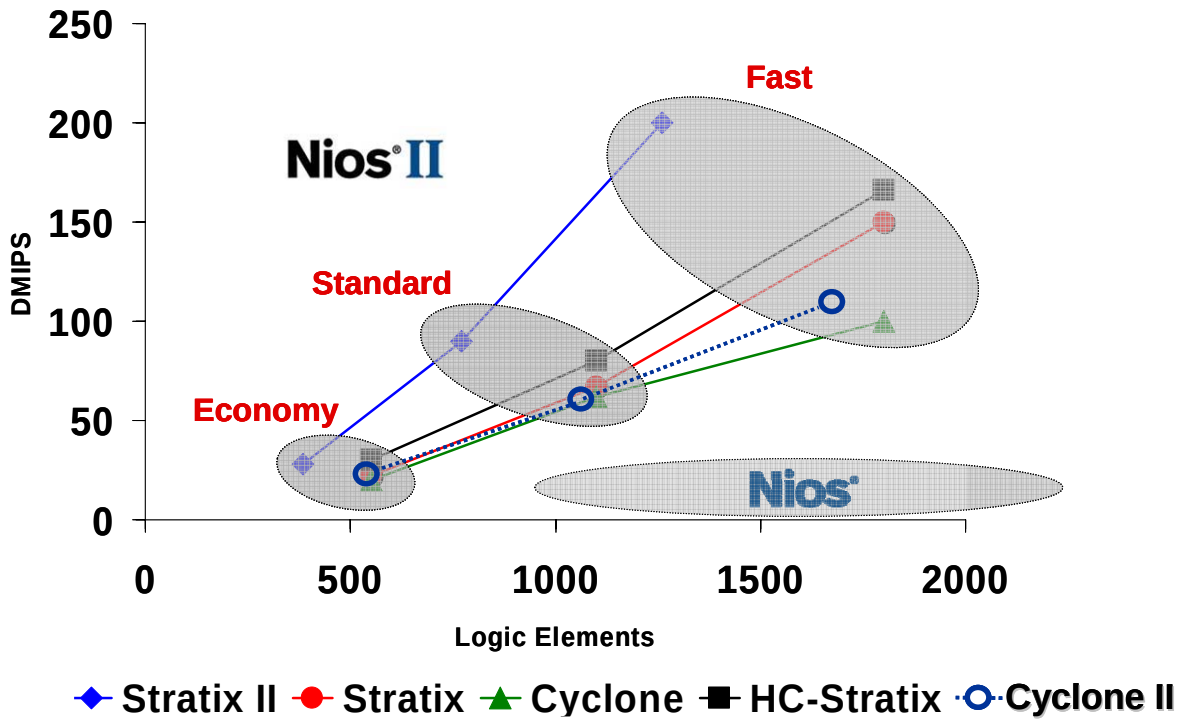


Figure 5 : Performances du processeur NIOS II sur différents circuits FPGA d'Altera

Dans la phase de construction du circuit SoPC (*System on Programmable Chip*) avec l'outil *SoPC Builder*, il est possible d'inclure différents périphériques standards en utilisant le bus d'adresses/données Altera *Avalon* :

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

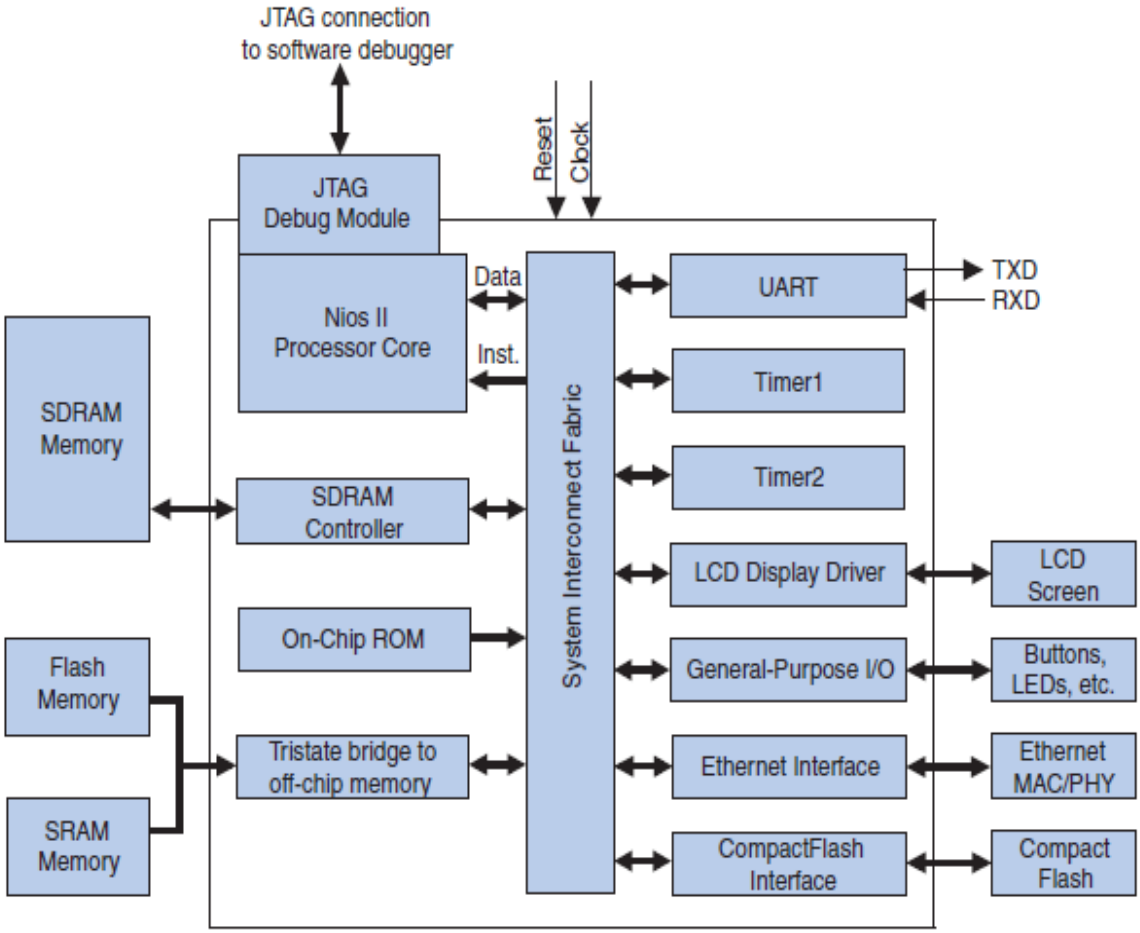


Figure 6 : Bus Avalon pour l'interfaçage des périphériques au processeur NIOS II

3. MISE EN ŒUVRE DE XENOMAI SUR LE PROCESSEUR SOFTCORE NIOS II

3.1. Carte cible

La carte choisie pour le portage de Xenomai sur le processeur NIOS II est une carte d'évaluation moyenne gamme d'Altera : carte Stratix 1S10.

Figure 7 : Carte cible Altera Stratix 1S10

La carte Stratix 1S10 possède les caractéristiques suivantes :

- Circuit FPGA Stratix II EP2S30F672C5.
- 1 Mo de SRAM 16 bits.
- 16 Mo de SDRAM 32 bits.
- 16 Mo de mémoire Flash.
- 1 support CompactFlash type I.
- 1 interface Ethernet 10/100 Mb/s.
- 2 ports série (RS-232 DB9).
- 2 ports d'extension pour carte fille maison.
- 1 connecteur JTAG.
- 4 boutons poussoirs.
- 8 leds utilisateurs.

- 2 afficheurs 7 segments.
- 1 afficheur LCD 2x16 caractères.

3.2. Configuration matérielle : construction du système SoPC

Il convient à l'aide de l'outil de synthèse *Quartus II* d'Altera de construire le système SoPC compatible avec le portage Xenomai sur processeur NIOS II. A l'issue de la synthèse, on récupèrera le fichier *.sof* de programmation du circuit FPGA de la carte cible Stratix 1S10.

❖ Design de référence

Le design de référence *Quartus II* pour Linux embarqué avec ou sans l'extension Temps Réel a été construit à partir du design *standard* fourni avec la carte Stratix 1S10 par Altera.

Ce design *standard* est fourni pour toutes les cartes Altera et est une bonne base de travail pour faire tourner dans un premier temps Linux embarqué.

La figure présente le design de référence *standard* :

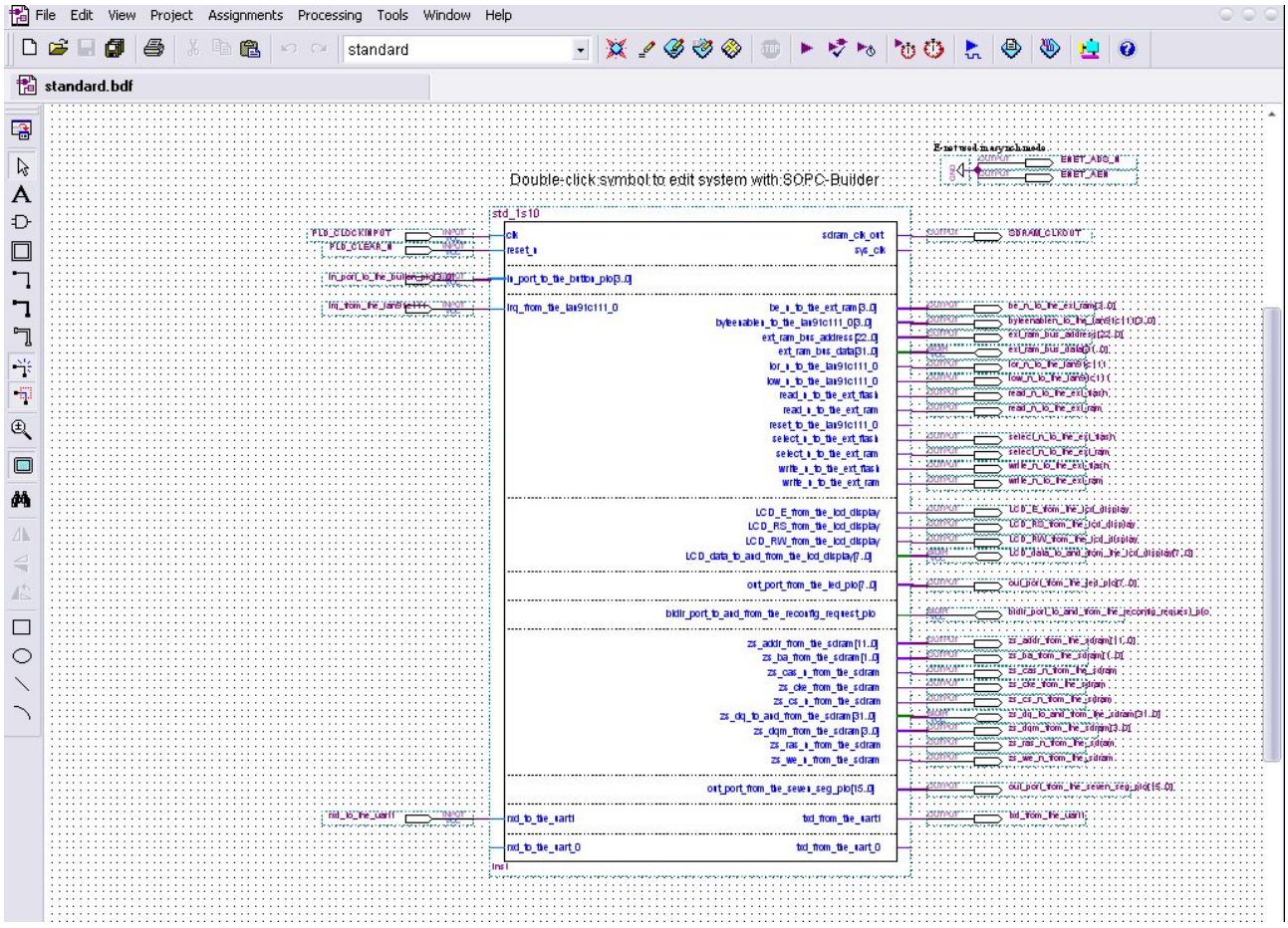
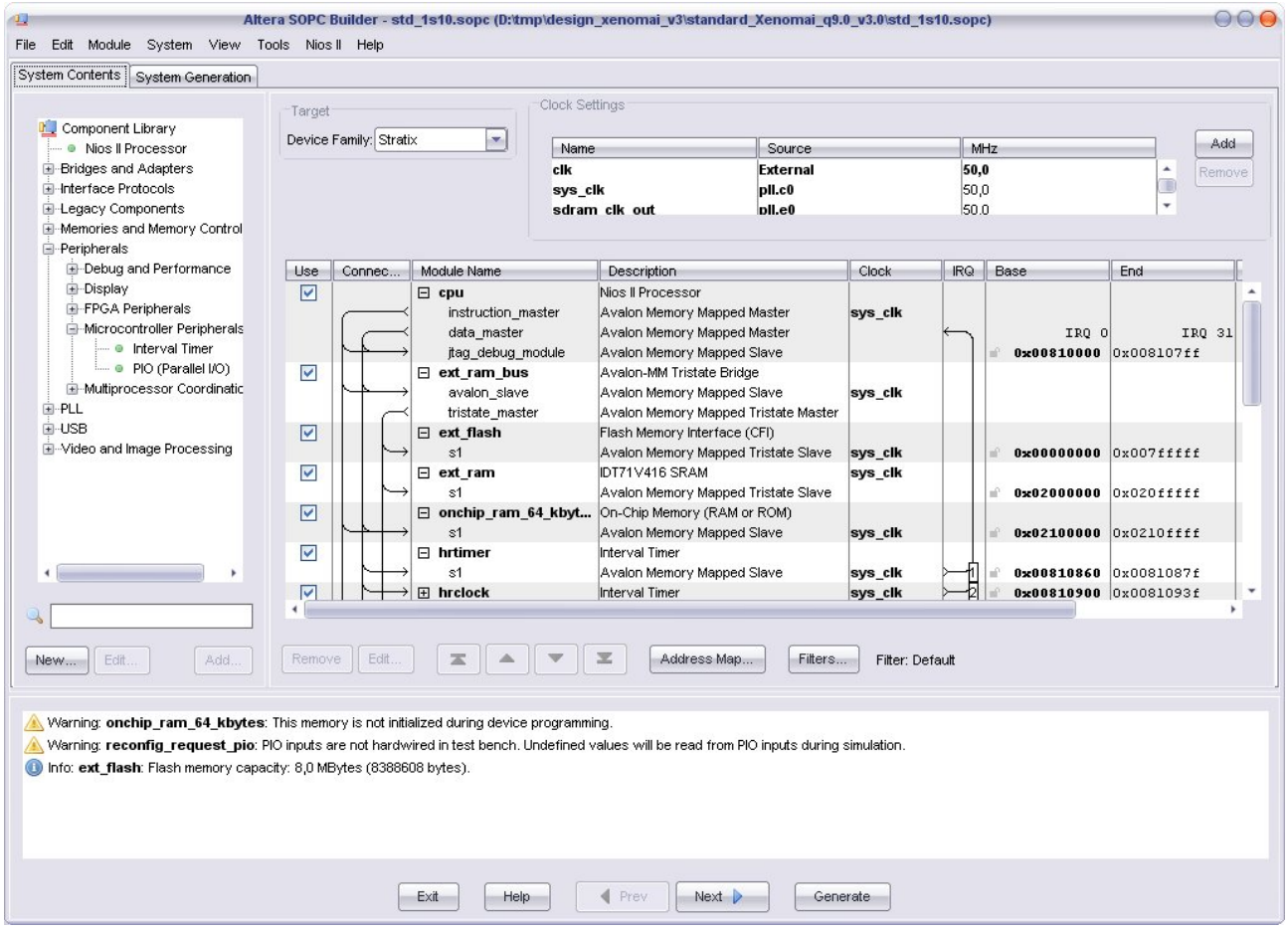
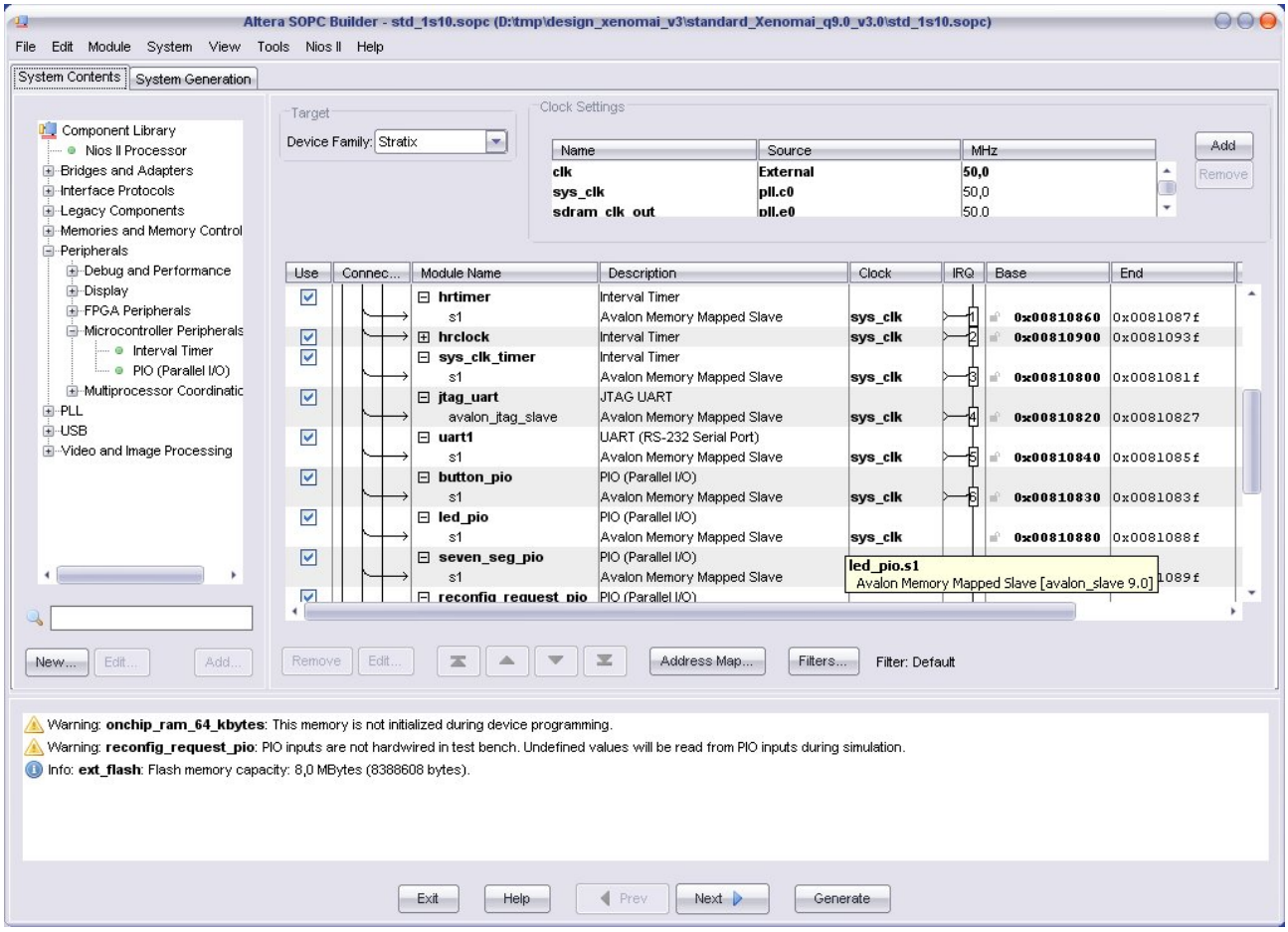


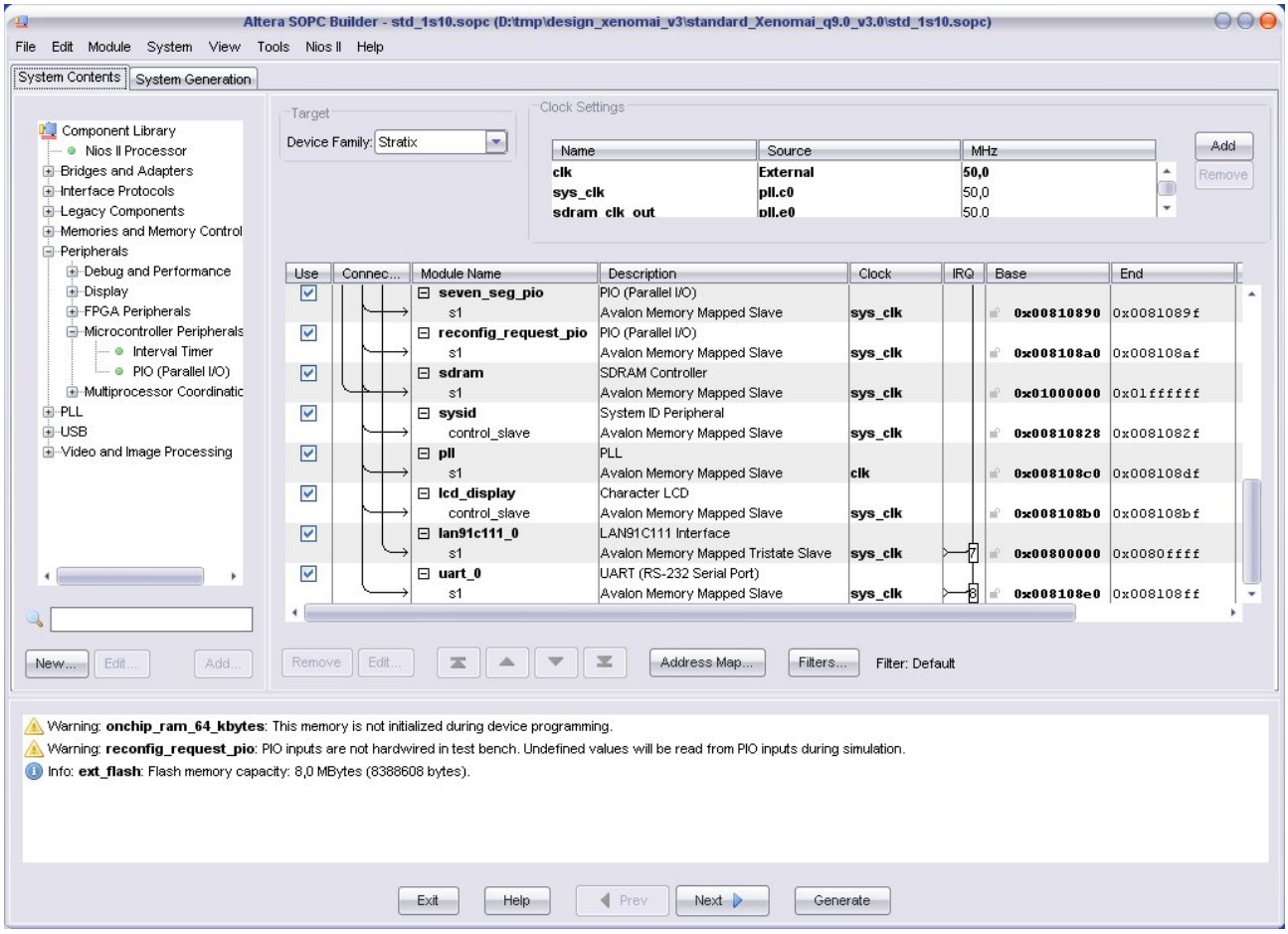
Figure 8 : Design de référence *standard Quartus II* de la carte cible Altera Stratix 1S10

Une fois le design de référence adopté, il convient de modifier le système SoPC en utilisant l'outil *SoPC Builder* d'Altera.

Les 3 figures suivantes montrent la configuration du système SoPC. De nouveaux timers ont été introduits dans le design pour le portage Xenomai mais n'interfèrent pas avec Linux embarqué.




Altera SOPC Builder - std_1s10.sopc (D:\tmp\design_xenomai_v3\standard_Xenomai_q9.0_v3.0\std_1s10.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Component Library
Nios II Processor
Bridges and Adapters
Interface Protocols
Legacy Components
Memories and Memory Control
Peripherals
Debug and Performance
Display
FPGA Peripherals
Microcontroller Peripherals
Interval Timer
PIO (Parallel I/O)
Multiprocessor Coordinatic
PLL
USB
Video and Image Processing
Target
Device Family: Stratix
Clock Settings
Name
Source
MHz
clk
External
50,0
sys_clk
pll.c0
50,0
sdram_clk_out
pll.c0
50,0
Add
Remove
Use
Connec...
Module Name
Description
Clock
IRQ
Base
End
hrtimer
Interval Timer
s1
Avalon Memory Mapped Slave
sys_clk
0x00810860
0x0081087f
hrclock
Interval Timer
sys_clk
0x00810900
0x0081093f
sys_clk_timer
Interval Timer
s1
Avalon Memory Mapped Slave
sys_clk
0x00810800
0x0081081f
jtag_uart
JTAG UART
avalon_jtag_slave
Avalon Memory Mapped Slave
sys_clk
0x00810820
0x00810827
uart1
UART (RS-232 Serial Port)
s1
Avalon Memory Mapped Slave
sys_clk
0x00810840
0x0081085f
button_pio
PIO (Parallel I/O)
s1
Avalon Memory Mapped Slave
sys_clk
0x00810830
0x0081083f
led_pio
PIO (Parallel I/O)
s1
Avalon Memory Mapped Slave
sys_clk
0x00810880
0x0081088f
seven_seg_pio
PIO (Parallel I/O)
s1
Avalon Memory Mapped Slave
led_pio.s1
Avalon Memory Mapped Slave [avalon_slave 9.0]
reconfig_request_pio
PIO (Parallel I/O)
New...
Edit...
Add...
Remove
Edit...
Address Map...
Filters...
Filter: Default
Warning: onchip_ram_64_kbytes: This memory is not initialized during device programming.
Warning: reconfig_request_pio: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Info: ext_flash: Flash memory capacity: 8,0 MBytes (8388608 bytes).
Exit
Help
Prev
Next
Generate



Figures 9 : Création des différents périphériques dans le système SoPC

Les points importants concernent l'attribution des IRQs et des plages d'adresses des registres des différents périphériques intégrés dans le système SoPC.

Il est important de respecter ces règles pour que le portage de Xenomai sur processeur NIOS II marche. Tous les numéros d'IRQs doivent être différents de 0 (autodétection au sens Linux sinon).

Type du périphérique	Nom du périphérique	Numéro d'IRQ	OS concerné
Timer 32 bits	*hrtimer*	1 *	Xenomai
Timer 64 bits	*hrclock*	2 *	Xenomai
Timer 32 bits	*sys_clk_timer*	3 *	Linux

Figure : Attribution des numéros des IRQs aux différents timers du système SoPC

* : l'ordre n'influence pas la priorité de l'IRQ avec le processeur NIOS II mais on a préféré regrouper les IRQs des timers…

❖ Mapping mémoire

Le mapping mémoire ne doit pas comporter de recouvrement dans l'attribution des plages d'adresses pour les registres des différents périphériques. Il n'y a pas de contraintes sinon, hormis les contraintes des différents types de mémoire de la carte cible.

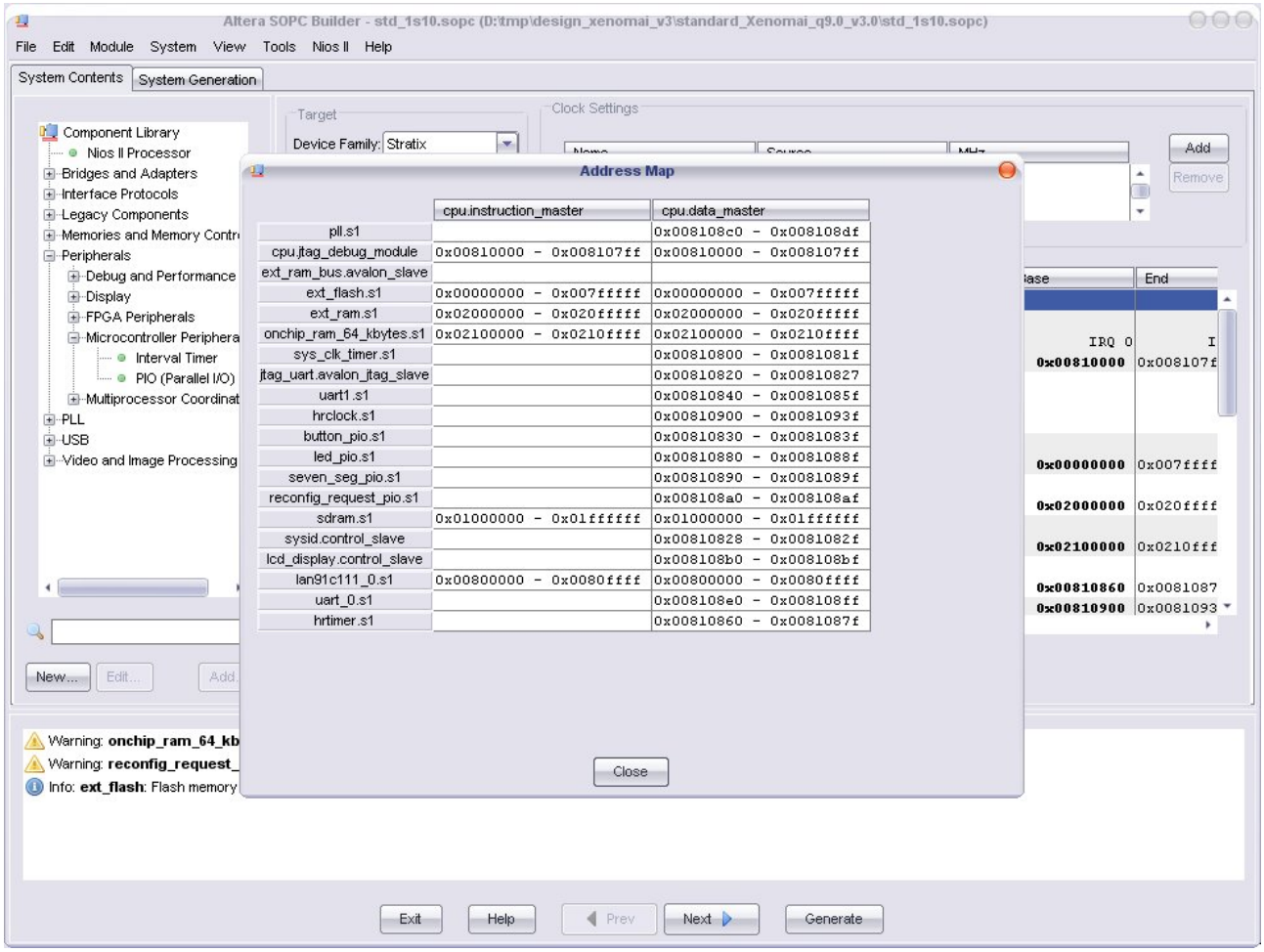


Figure 10 : Mapping mémoire du système SoPC

❖ Configuration du processeur NIOS II

Le processeur NIOS II est configuré au minimum dans sa version s (*standard*).

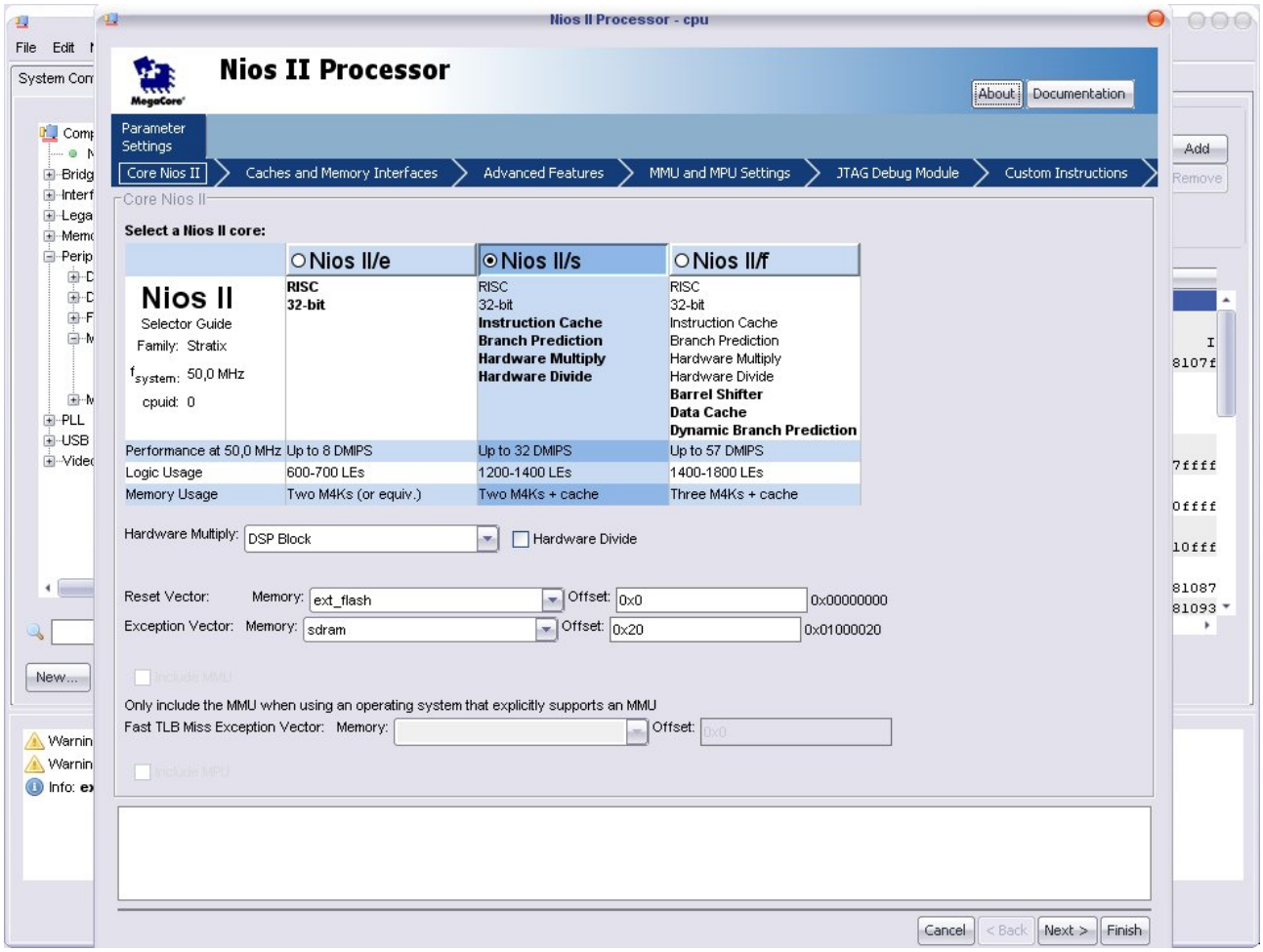


Figure 11 : Configuration du processeur NIOS II dans sa version s

Les caches d'instructions et de données sont validés :

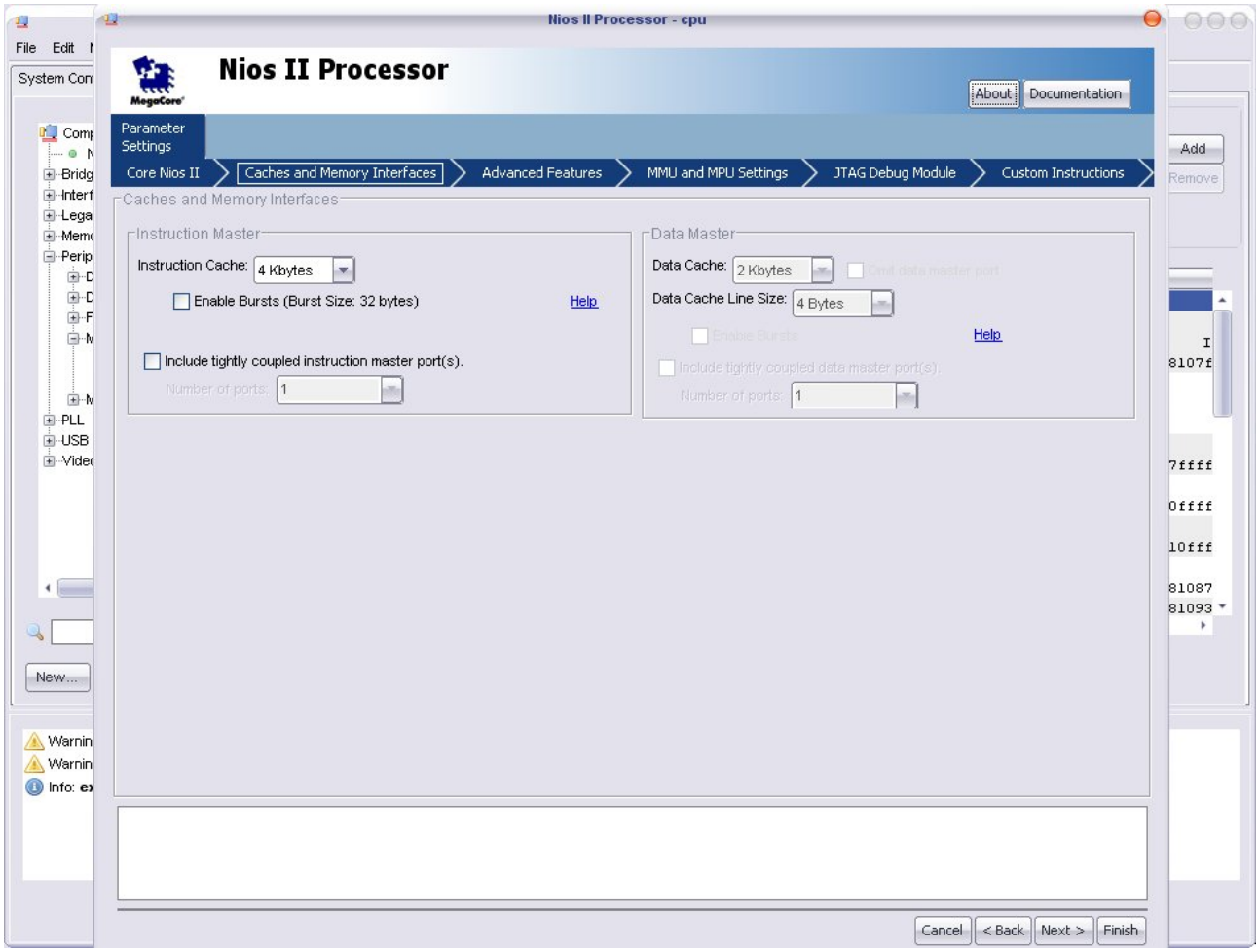
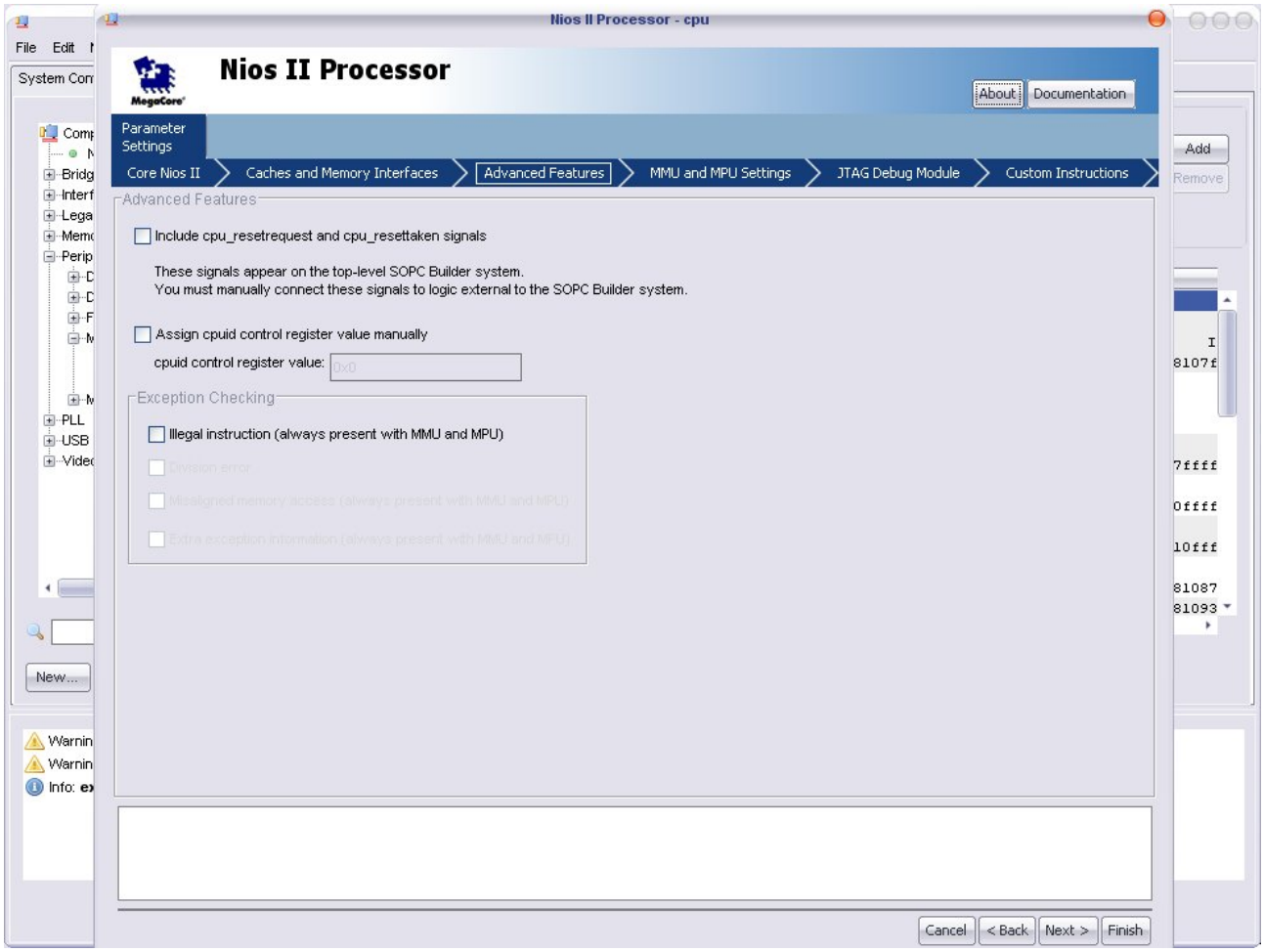
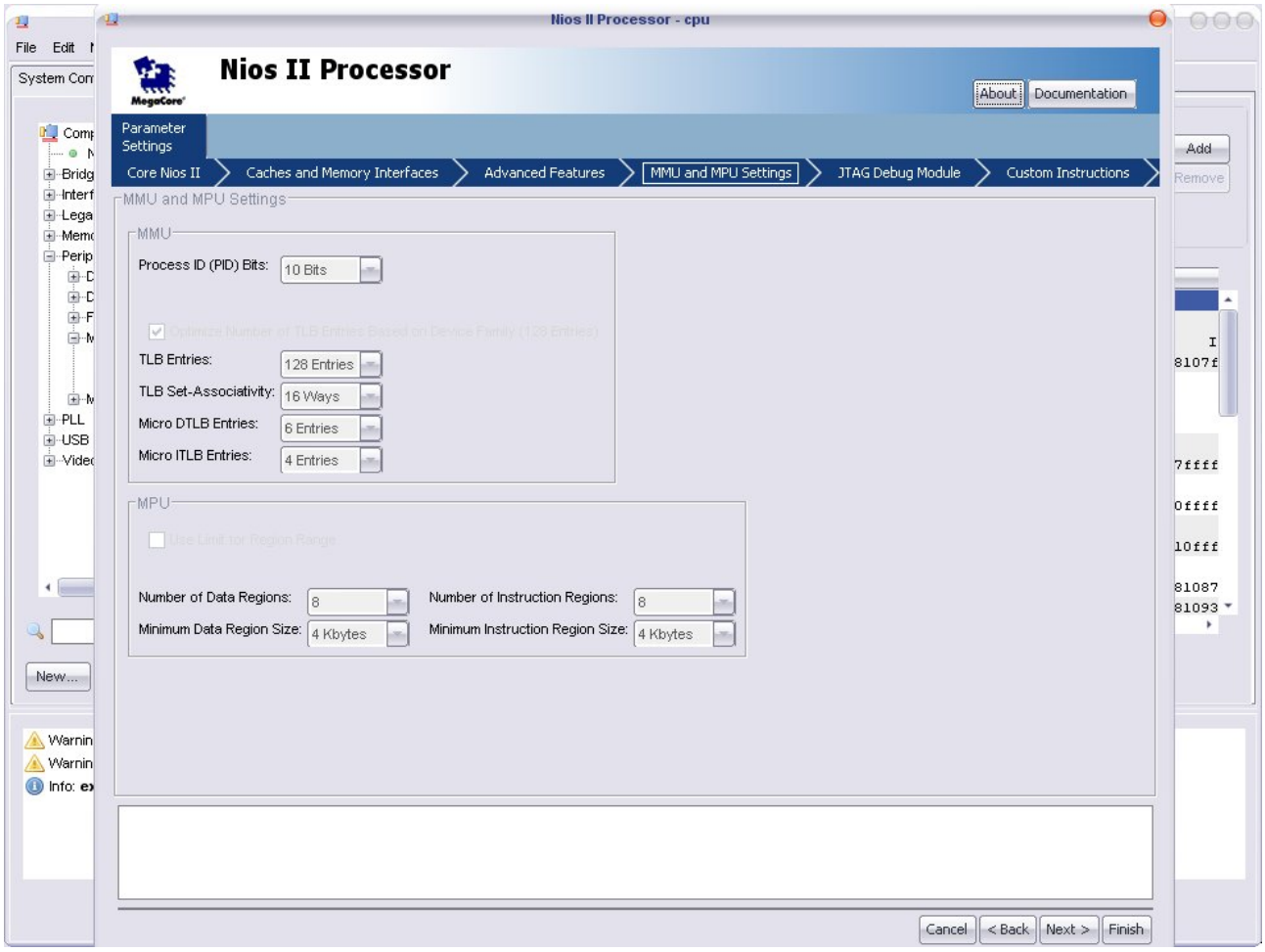


Figure 12 : Validation des caches d'instructions et de données du processeur NIOS II

Le circuit MMU ou MPU (apparus avec la version 8 de Quartus) n'est pas validé.

En conséquence, **on utilisera la version µClinux** pour la version Linux embarqué pour le processeur NIOS II avec ou sans l'usage de Xenomai…


Nios II Processor - cpu
Nios II Processor
About
Documentation
Parameter Settings
Core Nios II
Caches and Memory Interfaces
Advanced Features
MMU and MPU Settings
JTAG Debug Module
Custom Instructions
Advanced Features
Include cpu_resetrequest and cpu_resettaken signals
These signals appear on the top-level SOPC Builder system.
You must manually connect these signals to logic external to the SOPC Builder system.
Assign cpuid control register value manually
cpuid control register value:
0x0
Exception Checking
Illegal instruction (always present with MMU and MPU)
Division error
Misaligned memory access (always present with MMU and MPU)
Extra exception information (always present with MMU and MPU)
Cancel
< Back
Next >
Finish



Figures 13 : Circuits MMU et MPU dévalidés pour le processeur NIOS II

On validera le JTAG en version 1 pour le debug :

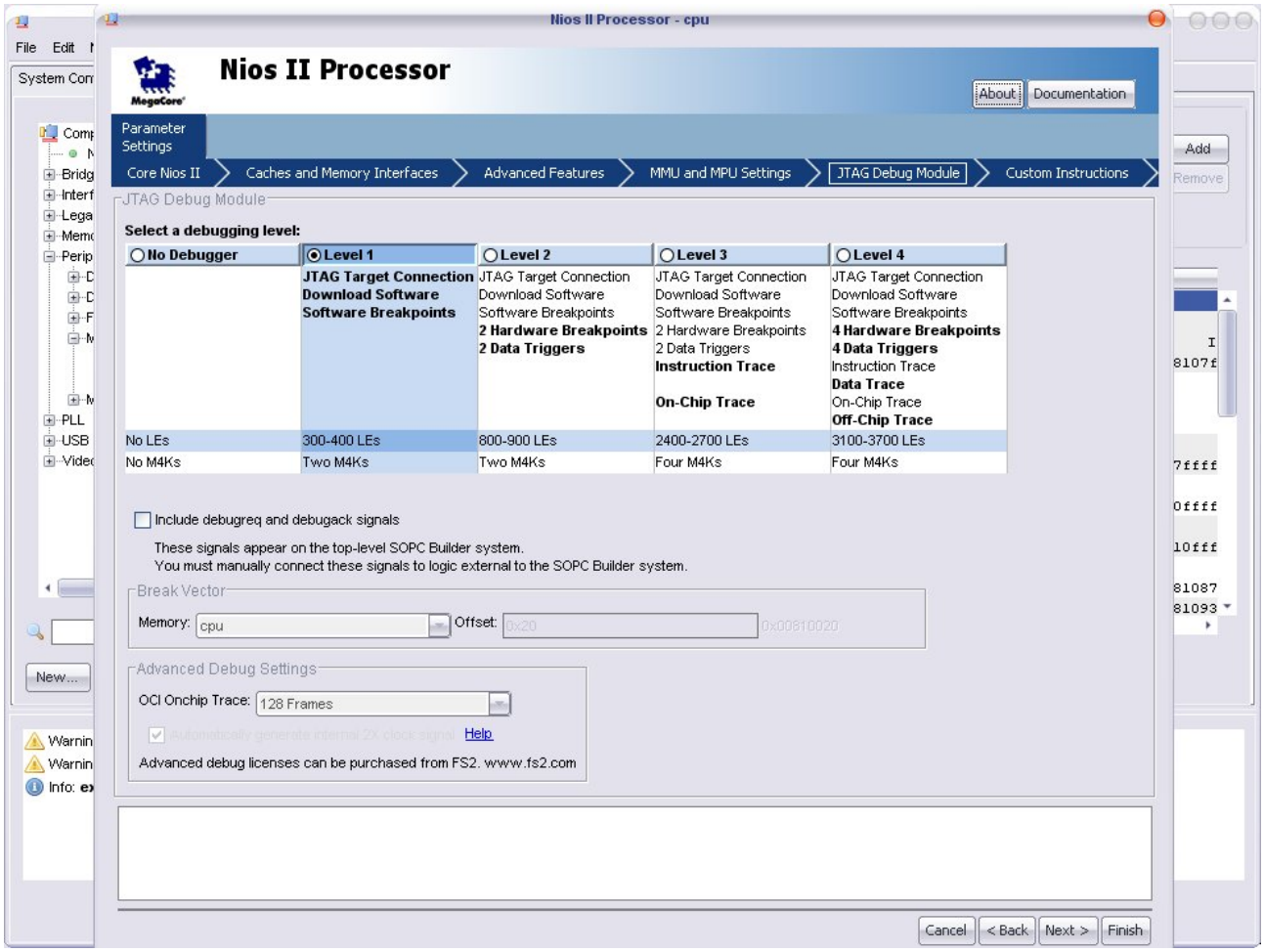


Figure 14 : Validation du JTAG (*level 1*)

On ne valide rien au niveau des *Custom Instructions* :

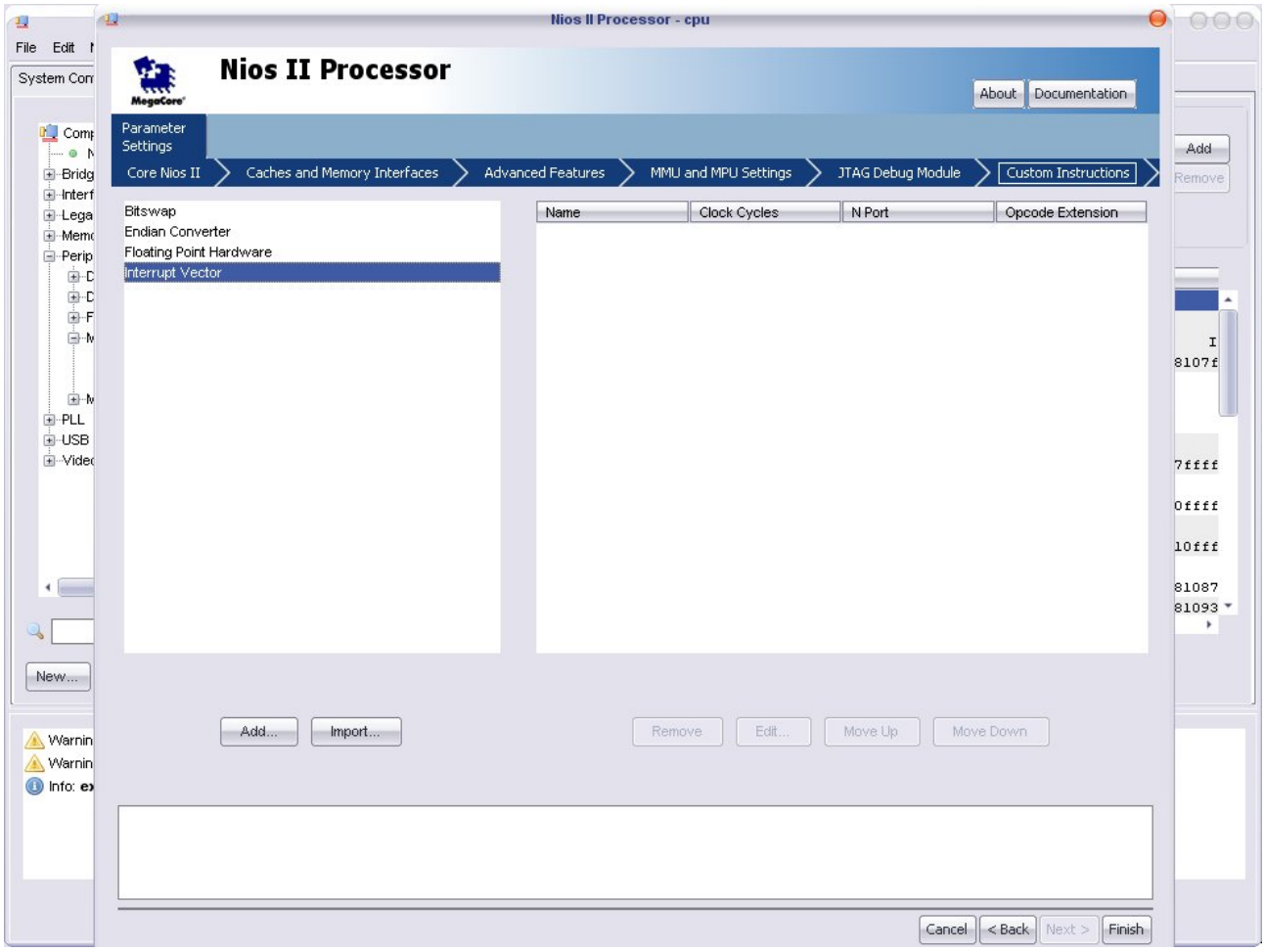


Figure 15 : Dévalidation des *Custom Instructions*

❖ Timer hrtimer

Le timer *hrtimer* est utilisé par Xenomai comme *tick timer*. Sa configuration dans *SoPC Builder* est la suivante :

- Timer : 32 bits.
- Période de timeout : 1 µs.
- Preset : custom. Writable period, readable snapshot, Start/Stop control bits.

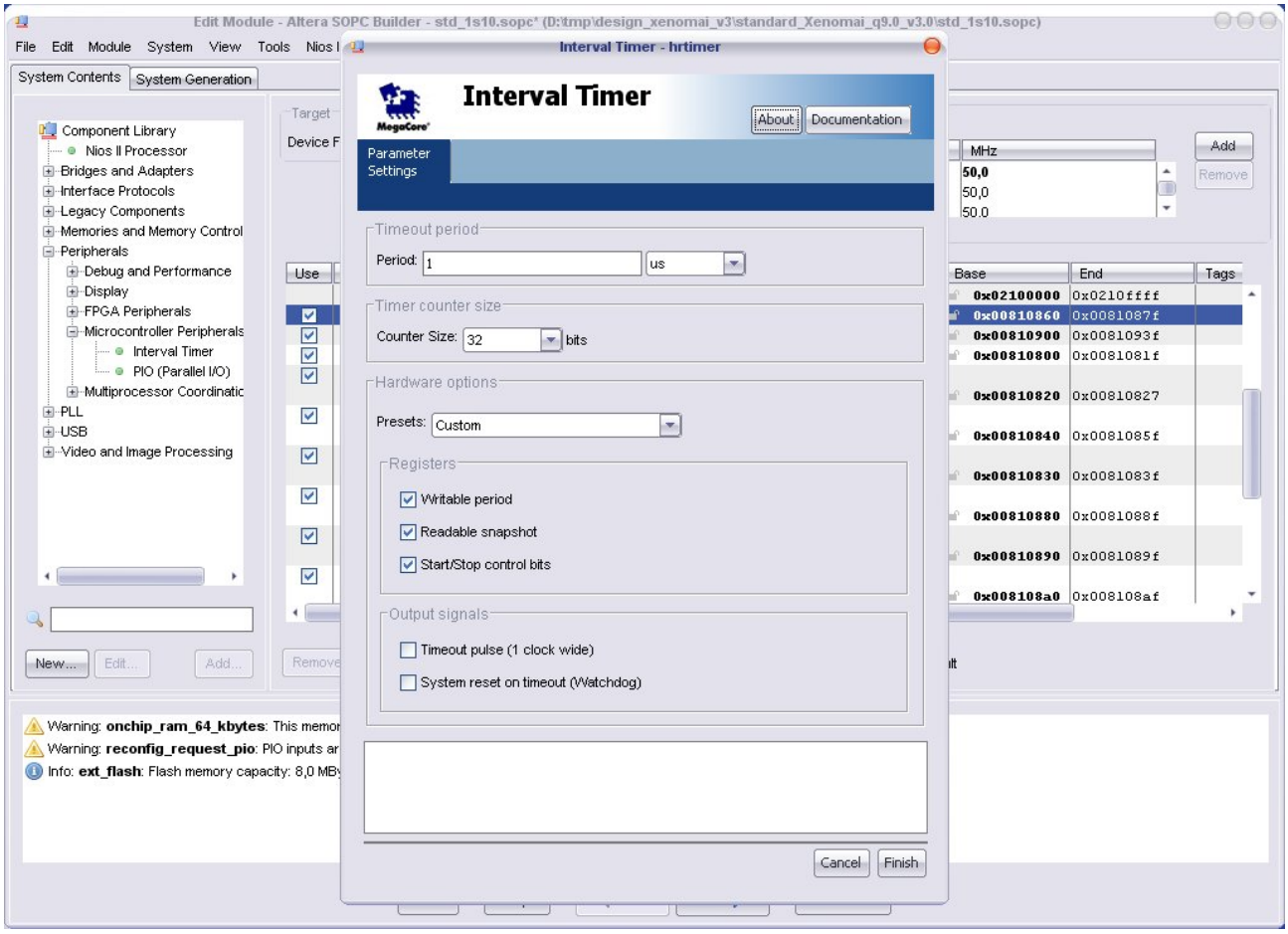


Figure 16 : Configuration du timer Xenomai *hrtimer*

❖ Timer hrclock

Le timer *hrclock* est utilisé par Xenomai comme mesure de temps par snapshot. Sa configuration dans *SoPC Builder* est la suivante :

- Timer : 64 bits.
- Période de timeout : 5 clocks. clock=20 ns, soit 100 ns. La fonctionnalité *timer* n'est pas utilisée par Xenomai.
- Preset : custom. Writable period, readable snapshot, Start/Stop control bits.

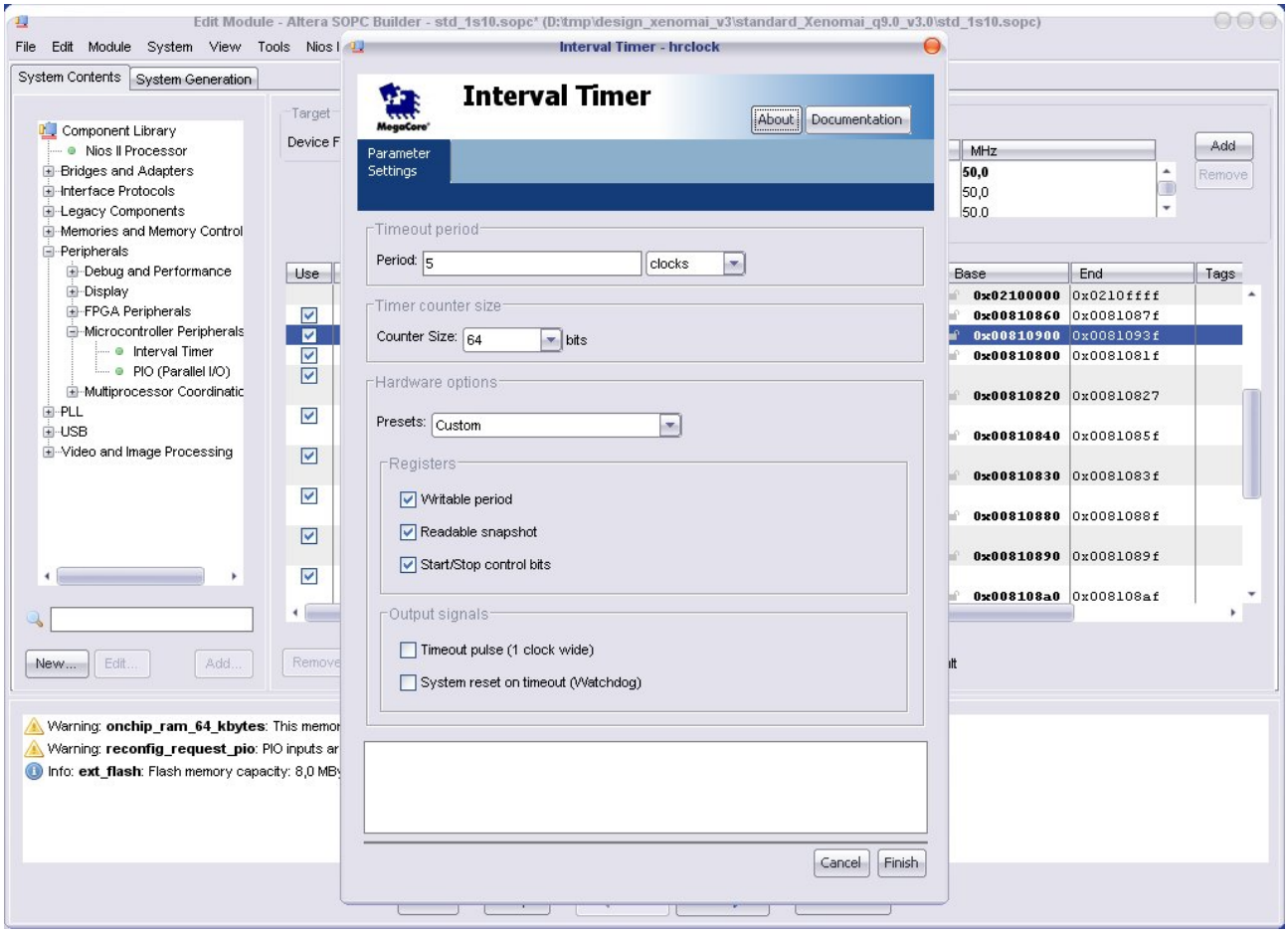


Figure 17 : Configuration du timer Xenomai mesure de temps *hrclock*

❖ Timer sys_clk_timer

REGLES IMPORTANTES DE CREATION DU SYSTEME SOPC POUR XENOMAI :

Quand on instancie les périphériques pour son système SoPC avec l'outil *SOPC Builder*, les périphériques apparaissent dans le fichier *.ptf* dans l'ordre de leur création.

Quand on crée le fichier *nios2.h* à partir du fichier *.ptf* , le parser utilisé (Perl script) cherche le premier timer Altera est l'utilise comme tick timer Linux.

On doit donc créer en premier le timer *sys_clk_timer* dnas le système SoPC et les autres timers en suivant.

Si vous n'avez pas fait cela, il faut effacer tous les timers Altera dans son système SoPC (*SOPC Builder*), enregistrer et créer à nouveau le timer *sys_clk_timer* !

Le timer *sys_clk_timer* est utilisé par Linux comme *tick timer*. Sa configuration dans *SoPC Builder* est la suivante :

- Timer : 32 bits.
- Période de timeout : 10 ms.
- Preset : custom. Writable period, readable snapshot, Start/Stop control bits.

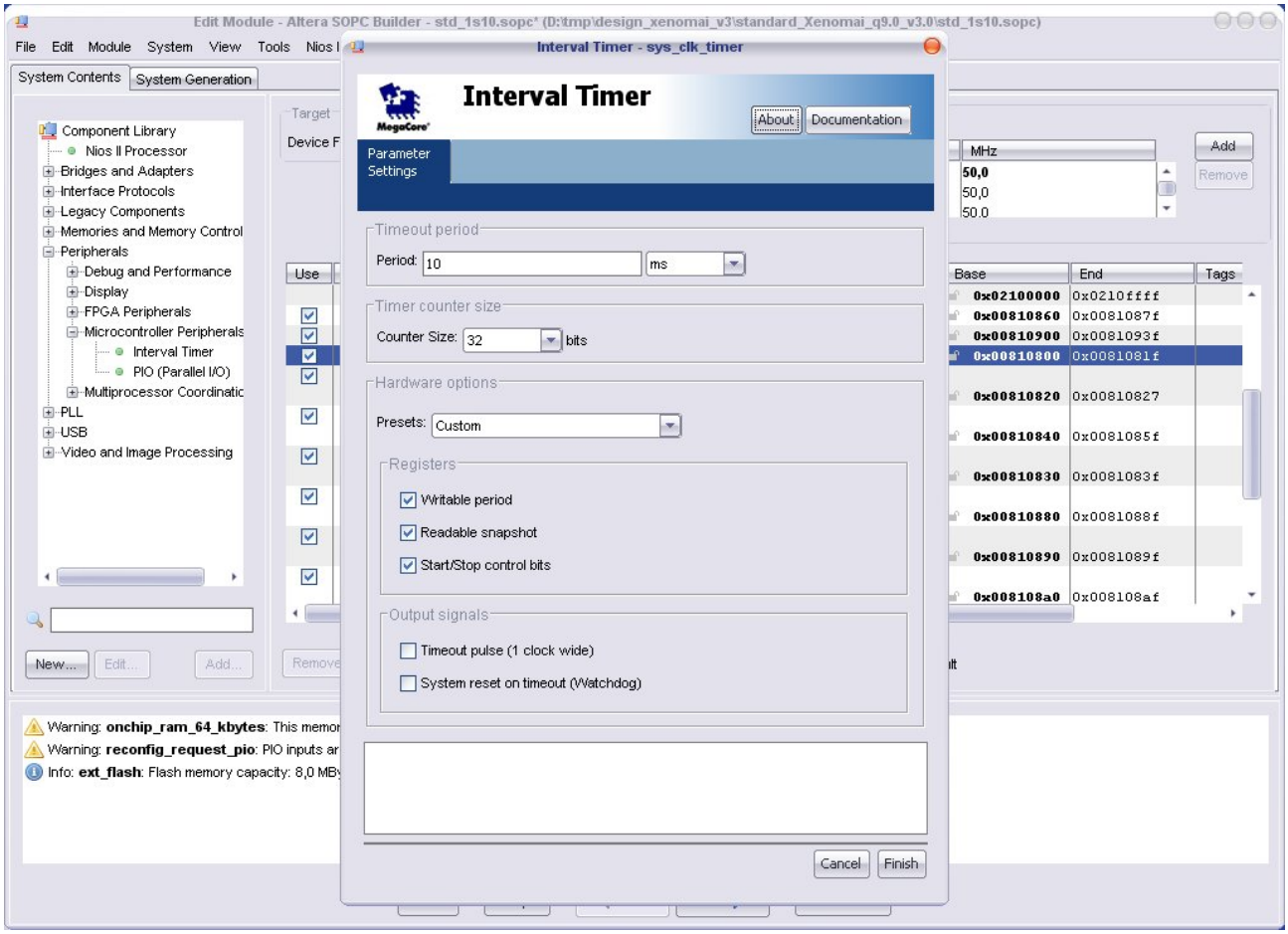


Figure 18 : Configuration du timer Linux *sys_clk_timer*

3.3. Installation des outils Altera sous Linux

Il est possible d'utiliser les outils Altera 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.

Le document [3] explique comment faire cette installation. La procédure a été testée sous Fedora 12.
On installera les outils Quartus II version 9.1 (à la date d'écriture de ce document) :

```
$ cd
$ wget ftp://ftp.altera.com/outgoing/release/91_quartus_linux.tar
$ wget ftp://ftp.altera.com/outgoing/release/91_nios2eds_linux.tar
$ wget ftp://ftp.altera.com/outgoing/release/91_modelsim_ae_linux.tar
$ tar -xvf 91_nios2eds_linux.tar
$ cd nios2eds
# ./install
$ cd
```

```
$ tar -xvf 91_modelsim_ae_linux.tar
$ cd modelsim_ae
# ./install
$ cd
$ tar -xvf 91_quartus_linux.tar
$ cd quartus
# ./install
$ cd
```

L'ensemble des logiciels est installé dans le répertoire */opt/altera9.1*.

On créera le script *n2sdk* que l'on placera sous *~/bin* :

```
!/bin/bash
# Run this for a Nios II SDK bash shell
export LM_LICENSE_FILE=1700@localhost
SOPC_KIT_NIOS2=/opt/altera9.1/nios2eds
export SOPC_KIT_NIOS2
SOPC_BUILDER_PATH_91=/opt/altera9.1/nios2eds
export SOPC_BUILDER_PATH_91
unset GCC_EXEC_PREFIX
QUARTUS_ROOTDIR=/opt/altera9.1/quartus
export QUARTUS_ROOTDIR
export PERL5LIB=/usr/lib/perl5/5.10.0
bash --rcfile $QUARTUS_ROOTDIR/sopc_builder/bin/nios_bash
```

Pour pouvoir utiliser les outils Altera *Quartus II* et *ModelSim*, il faut bien sûr avoir une licence Altera 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 le JTAG pour le module USB Blaster de la carte cible :

```
# mkdir /etc/jtagd
# cp /opt/altera9.1/quartus/linux/pgm_parts.txt /etc/jtagd/jtagd.pgm_parts
$ n2sdk
[NiosII EDS]$ su
Mot de passe :
# jtagd
# exit
[NiosII EDS]$ touch ~/.jtag.conf
[NiosII EDS]$ jtagconfig
1) USB-Blaster [USB 2-1.3.2]
  020010DD   EP1S10
```

On vérifie que la commande *nios2-download* fonctionne pour télécharger un fichier dans la cible par le JTAG:

```
[NiosII EDS]$ nios2-download
Using cable "USB-Blaster [USB 2-1.3.2]", device 1, instance 0x00
Pausing target processor: OK
Restarting target processor
[NiosII EDS]$
```

Les outils Altera sont opérationnels sous Linux.

3.4. Configuration logicielle : configuration et compilation de Linux pour NIOS II

Le processeur NIOS II étant configuré sans MMU ou MPU, la distribution Linux embarqué µClinux est utilisée.

Il convient dans un premier temps de télécharger la distribution µClinux pour NIOS II. On se réfèrera au document [3] pour plus de détails :

```
$ cd
$ wget http://www.niosftp.com/pub/uclinux/nios2-linux-20090730.tar
```

On décompresse l'archive :

```
$ tar -xvf nios2-linux-20090730.tar
```

On se place dans le répertoire nios2-linux :

```
$ cd nios2-linux
```

On lance le script *checkout* pour récupérer les différents fichiers sources et la mise à jour par le script *update* :

```
$ ./checkout
$ ./update
```

On installe le compilateur croisé *gcc* pour le processeur NIOS II :

```
$ wget http://www.niosftp.com/pub/gnutools/nios2gcc-20080203.tar.bz2
# tar jxf nios2gcc-20080203.tar.bz2 -C /
```

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

```
PATH=$PATH:/opt/nios2/bin
```

On vérifie que le compilateur gcc pour NIOS II est opérationnel :

```
$ nios2-linux-gcc -v
Reading specs from /opt/nios2/lib/gcc/nios2-linux-uclibc/3.4.6/specs
Configured with: /root/buildroot/toolchain_build_nios2/gcc-3.4.6/configure --prefix=/opt/nios2 --build=i386-pc-linux-gnu --host=i386-pc-linux-gnu --target=nios2-linux-uclibc --enable-languages=c,c++ --disable-__cxa_atexit --enable-target-optspace --with-gnu-ld --disable-shared --disable-nls --enable-threads --enable-multilib
Thread model: posix
gcc version 3.4.6
```

On suppose maintenant que *$uClinux_dist* représente le chemin du répertoire des fichiers sources de µClinux, c'est à dire ici *nios2-linux/uClinux-dist*.
On configure µClinux pour le processeur NIOS II :

```
$ cd $uClinux_dist
$ make menuconfig
```

Dans les écrans menuconfig, on vérifie que les options suivantes sont bien validées :

```
Vendor/Product Selection --->
    --- Select the Vendor you wish to target
        Vendor (Altera)  --->
    --- Select the Product you wish to target
        Altera Products (nios2)   --->

Kernel/Library/Defaults Selection --->
    --- Kernel is linux-2.6.x
        Libc Version (None)  --->
    [*] Default all settings (lose changes)
    [ ] Customize Kernel Settings
    [ ] Customize Vendor/User Settings
    [ ] Update Default Vendor Settings
```

On fait le lien entre le système matériel SoPC et µClinux via le fichier *.ptf* de son design afin de générer le fichier header *nios2.h* :

```
$ make vendor_hwselect SYSPTF=/chemin_vers_son_design/son_design.ptf
```

Par exemple :

```
$ make vendor_hwselect SYSPTF=~/design_xenomai/std_1s10.ptf
RUNNING hwselect
--- Please select which CPU you wish to build the kernel against:
(1) cpu - Class: altera_nios2 Type: s Version: 7.080900

Selection: 1

--- Please select a device to execute kernel from:

(1) ext_flash
     Class: altera_avalon_cfi_flash
     Size: 8388608 bytes

(2) onchip_ram_64_kbytes
     Class: altera_avalon_onchip_memory2
     Size: 65536 bytes

(3) ext_ram
     Class: altera_nios_dev_kit_stratix_edition_sram2
     Size: 1048576 bytes

(4) sdram
     Class: altera_avalon_new_sdram_controller
     Size: 16777216 bytes

Selection: 4

--- Summary using
PTF: /home/kadionik/design_xenomai/std_1s10.ptf
CPU:                            cpu
Program memory to execute from: sdram
```

On configurera µClinux pour utiliser plutôt la liaison série de la carte cible pour la console Linux au lieu du JTAG émulé en liaison série :

```
$ make menuconfig
Menu Device Driver>Character devices>Serial drivers
    [ ] 8250/16550 and compatible serial support
        *** Non-8250 serial port support ***
    [ ] Altera JTAG UART support
    [*] Altera UART support
    (4)   Maximum number of Altera UART ports
    (115200) Default baudrate for Altera UART ports
    [*]   Altera UART console support
```

On compile le noyau µClinux :

```
$ make
```

On télécharge ensuite le fichier *zImage* dans la carte cible par le JTAG :

```
$ n2sdk
[NiosII EDS]$ nios2-download -g images/zImage
```

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 µClinux :

```
$ minicom
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.6.30 (kadionik@linux00) (gcc version 3.4.6) #4 PREEMPT Thu Nov 9


uClinux/Nios II
Built 1 zonelists in Zone order, mobility grouping off.  Total pages: 4064
Kernel command line:
PID hash table entries: 64 (order: 6, 256 bytes)
Dentry cache hash table entries: 2048 (order: 1, 8192 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory available: 11988k/4110k RAM, 0k/0k ROM (1928k kernel code, 2181k data)
Calibrating delay loop... 24.06 BogoMIPS (lpj=120320)
Mount-cache hash table entries: 512
net_namespace: 264 bytes
NET: Registered protocol family 16
init_BSP(): registering device resources
bio: create slab <bio-0> at 0
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 512 (order: 0, 4096 bytes)
TCP bind hash table entries: 512 (order: -1, 2048 bytes)
TCP: Hash tables configured (established 512 bind 512)
TCP reno registered
NET: Registered protocol family 1
io scheduler noop registered
io scheduler deadline registered (default)
ttyS0 at MMIO 0x810840 (irq = 5) is a Altera UART
console [ttyS0] enabled
ttyS1 at MMIO 0x8108e0 (irq = 8) is a Altera UART
smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0: SMC91C11xFD (rev 1) at 80800300 IRQ 7 [nowait]
eth0: Invalid ethernet MAC address.  Please set using ifconfig
dm9000 Ethernet Driver, V1.31
TCP cubic registered
```

```
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
Freeing unused kernel memory: 1868k freed (0x1230000 - 0x1402000)
Shell invoked to run file: /etc/rc
Command: hostname uClinux
Command: mount -t proc proc /proc -o noexec,nosuid,nodev
Command: mount -t sysfs sysfs /sys -o noexec,nosuid,nodev
Command: mount -t devpts devpts /dev/pts -o noexec,nosuid
Command: mount -t usbfs none /proc/bus/usb
mount: mounting none on /proc/bus/usb failed: No such file or directory
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: cat /etc/motd
Welcome to
          ____ _  _
         /  __| ||_|
   _    _| |  | | _ ____  _   _  _  _
  | | | | |  | || |   _ \| | | |\ \/ /
  | |_| | |__| || | | | | | |_| |/    \
  |  ___\____|_||_|_| |_|\____|\_/\_/
  | |
  |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
```

3.5. Configuration logicielle : configuration et compilation de Xenomai pour NIOS II

Il convient de télécharger Xenomai depuis le site officiel par *git* :

```
$ cd
$ git clone git://xenomai.org/xenomai-head.git
```

On synchronise ensuite le noyau µClinux avec la version (*commit*) du patch *ipipe* pour NIOS II (lire le fichier `ksrc/nios2/patches/README`) :

```
$ cd
$ cd nios2-linux/linux-2.6
$ git checkout -b ipipe d01303a1035a39e445007c7522d89ad985c4153c
```

On suppose maintenant que *$xenomai_root* représente le chemin du répertoire des fichiers sources de Xenomai, c'est à dire ici *xenomai-head*.
Il convient ensuite d'appliquer le patch ipipe pour le processeur NIOS II sur les sources du noyau µClinux :

```
$ cd
$ cd $xenomai_root
```

```
$ export linux_tree=$HOME/nios2-linux/linux-2.6
$ ./scripts/prepare-kernel.sh --arch=nios2 \
--adeos=$xenomai_root/ksrc/arch/nios2/patches/adeos-ipipe-2.6.30-nios2-* \
--linux=$linux_tree
```

On compile alors les outils Xenomai pour les intégrer dans le *romfs* de µClinux :

```
$ ./configure –host=nios2-linux
$ make install DESTDIR=$uClinux_dist/romfs
```

On supprimera toute la partie documentation de Xenomai dans le *romfs* :

```
$\rm -rf $uClinux_dist/romfs/usr/xenomai/share/doc
$\rm -rf $uClinux_dist/romfs/usr/xenomai/share/man
```

On recompile enfin le noyau µClinux intégrant Xenomai :

```
$ cd $uClinux_dist
$ make
```

On télécharge ensuite le fichier *zImage* dans la carte cible par le JTAG :

```
$ n2sdk
[NiosII EDS]$ nios2-download -g images/zImage
```

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 µClinux :

```
$ minicom
Uncompressing Linux... Ok, booting the kernel.
Linux version 2.6.30 (kadionik@linux01) (gcc version 3.4.6) #18 PREEMPT Tue Mar0


uClinux/Nios II
Built 1 zonelists in Zone order, mobility grouping off.  Total pages: 4064
Kernel command line:
NR_IRQS:32
PID hash table entries: 64 (order: 6, 256 bytes)
I-pipe 1.1-00: pipeline enabled.
Dentry cache hash table entries: 2048 (order: 1, 8192 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory available: 11572k/4525k RAM, 0k/0k ROM (1931k kernel code, 2593k data)
Calibrating delay loop... 24.06 BogoMIPS (lpj=120320)
Mount-cache hash table entries: 512
net_namespace: 264 bytes
NET: Registered protocol family 16
init_BSP(): registering device resources
bio: create slab <bio-0> at 0
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 512 (order: 0, 4096 bytes)
TCP bind hash table entries: 512 (order: -1, 2048 bytes)
TCP: Hash tables configured (established 512 bind 512)
TCP reno registered
NET: Registered protocol family 1
I-pipe: Domain Xenomai registered.
Xenomai: hal/nios2 started.
```

```
Xenomai: scheduling class idle registered.
Xenomai: scheduling class rt registered.
Xenomai: real-time nucleus v2.5.2 (Souls Of Distortion) loaded.
Xenomai: starting native API services.
Xenomai: starting POSIX services.
Xenomai: starting RTDM services.
io scheduler noop registered
io scheduler deadline registered (default)
ttyS0 at MMIO 0x810840 (irq = 5) is a Altera UART
console [ttyS0] enabled
ttyS1 at MMIO 0x8108e0 (irq = 8) is a Altera UART
smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0: SMC91C11xFD (rev 1) at 80800300 IRQ 7 [nowait]
eth0: Invalid ethernet MAC address.  Please set using ifconfig
dm9000 Ethernet Driver, V1.31
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
Freeing unused kernel memory: 2276k freed (0x1232000 - 0x146a000)
Shell invoked to run file: /etc/rc
Command: hostname uClinux
Command: mount -t proc proc /proc -o noexec,nosuid,nodev
Command: mount -t sysfs sysfs /sys -o noexec,nosuid,nodev
Command: mount -t devpts devpts /dev/pts -o noexec,nosuid
Command: mount -t usbfs none /proc/bus/usb
mount: mounting none on /proc/bus/usb failed: No such file or directory
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: cat /etc/motd
Welcome to


          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
/> cd /usr/xenomai/bin
/usr/xenomai/bin> ./latency -t2
== Sampling period: 10000 us
== Test mode: in-kernel timer handler
== All results in microseconds
warming up...
RTT|  00:00:01  (in-kernel timer handler, 10000 us period, priority 99)
RTH|-----lat min|-----lat avg|-----lat max|-overrun|----lat best|---lat worst
RTD|      73.080|      83.223|      90.020|       0|      73.080|      90.020
RTD|       1.820|      55.164|     104.840|       0|       1.820|     104.840
RTD|       3.240|      65.454|     106.160|       0|       1.820|     106.160
```

```
RTD|       0.920|      63.167|     100.800|       0|       0.920|     106.160
RTD|       1.160|      65.735|     103.860|       0|       0.920|     106.160
RTD|      -0.600|      62.741|     105.180|       0|      -0.600|     106.160
RTD|       3.520|      64.406|     105.480|       0|      -0.600|     106.160
RTD|       1.720|      65.379|     105.100|       0|      -0.600|     106.160
RTD|       2.280|      66.060|     106.980|       0|      -0.600|     106.980
RTD|       4.900|      65.314|     106.320|       0|      -0.600|     106.980
```

4. CONCLUSION

Le portage de Xenomai pour le processeur NIOS II dans l'environnement µClinux (sans MMU) est opérationnel.

Le portage est fonctionnel sur les cartes cibles suivantes et un design de référence Xenomai est disponible sur le site du projet [5] :

- Altera Stratix 1S10 fonctionnant à 50 MHz.
- Altera Cyclone 3C25 fonctionnant à 100 MHz.
- Altera Cyclone 3C120 fonctionnant à 100 MHz.

5. REFERENCES DOCUMENTAIRES

[1] Projet Xenomai. http://www.xenomai.org

[2] Projet µClinux pour processeur NIOS II. http://www.alterawiki.com/

[3] Page InstallNios2Linux. http://www.alterawiki.com/wiki/InstallNios2Linux

[4] Page QuartusforLinux. http://www.alterawiki.com/wiki/Quartus_for_Linux

[5] Page des ressources ENSEIRB-MATMECA de Linux embarqué et du portage de Xenomai sur processeur NIOS II. http://www.enseirb.fr/~kadionik/nios-xenomai

6. REFERENCES LOGICIELLES :

[1] Outils Altera sous Linux : ftp://ftp.altera.com/outgoing/release

[2] Portage µClinux pour le processeur NIOS II : ftp://ftp.altera.com/outgoing/nios2-linux-20080619.tar

[3] Compilateur croisé gcc pour processeur NIOS II : http://www.niosftp.com/pub/gnutools/nios2gcc-20080203.tar.bz2

[4] Portage Xenomai pour le processeur NIOS II : http://git.xenomai.org/

[5] Ces briques logicielles sont en secours sur la page Xenomai pour le processeur NIOS II de l'ENSEIRB-MATEMCA : http://www.enseirb.fr/~kadionik/nios-xenomai/