

Using Impulse C With Xilinx® Open Source Linux on a Virtex™-5 PowerPC

Platform Support Package (PSP) and Tutorial

Mei Xu

Impulse Accelerated Technologies, Inc.

Copyright© 2009 Impulse Accelerated Technologies, Inc.

Platform Support Package Details

The Impulse C Platform Support Package (PSP) "Xilinx Open Source Linux Virtex-5 APU" has been adapted from the Virtex-5 APU PSP. The HDL hardware side is unchanged from the original, standalone PSP; only the software side is modified in support of embedded Linux. An alternate "Xilinx Open Source Linux Virtex-4 APU" PSP is also ready to use with Virtex-4 FPGA devices.

This tutorial assumes that you have some knowledge of Impulse C, and of the Xilinx Virtex-5 FX devices. Experience with creating standalone Impulse C applications (with no embedded Linux) on the Xilinx ML507 or equivalent board is also helpful.

In the standalone mode APU PSP, `co_streams` are implemented using Xilinx APU specific instructions `stwfcmx` and `lwfcmx`. When running Linux over PowerPC, we replace these instructions with AltiVec extended instructions `stvewx` and `lvewx`.

In `co_stream.c`, the functions `co_stream_read`, `co_stream_write` and `co_stream_close` are supported. The primitives `HW_STREAM_READ`, `HW_STREAM_WRITE` and `HW_STREAM_CLOSE` are supported as well.

When the PowerPC CPU runs applications in standalone mode, the application software can have privileged instructions, such as `mtmsr` to enable the APU on the MSR register. When an application runs on Linux user space, however, the `mtmsr` instruction is not allowed. Here, then, we must modify some kernel code to allow the APU to work.

In the directory `linux-2.6-xlnx/arch/powerpc/include/asm`, you must modify header files `reg.h` and `reg_booke.h` as follows:

```
----- reg.h -----
```

```

#ifdef CONFIG_PPC64
.....
#else /* 32-bit */
/* Default MSR for kernel mode. */
#ifndef MSR_KERNEL /* reg_booke.h also defines this */
#define MSR_KERNEL (MSR_ME|MSR_RI|MSR_IR|MSR_DR|MSR_VEC)
#endif

----- reg_booke.h -----

/* Default MSR for kernel mode. */
#if defined (CONFIG_40x)
#define MSR_KERNEL
(MSR_ME|MSR_RI|MSR_IR|MSR_DR|MSR_CE|MSR_VEC)
#elif defined(CONFIG_BOOKE)
#define MSR_KERNEL (MSR_ME|MSR_RI|MSR_CE|MSR_VEC)
#endif
-----

```

The basic idea is to add the "MSR_VEC" bit to "MSR_KERNEL" for APU activation.

In the PSP, "export.tcl" has been modified so that a new directory "user_app" is constructed, and application code is copied into it. The EDK/code directory is no longer included, since the source files won't be needed in EDK. The necessary Impulse library source files are being copied to the sub-directory "libImpulse" with a Makefile. "genMakefile.tcl" is added to the PSP, so that a Makefile for the software application is generated and copied into the "user_app" folder.

You need to go to the "user_app/libImpulse" directory to build the Impulse library, and then go up to the "user_app" directory to build your software application. A Makefile is already generated for you, so just a "make" command will do the job.

In "co.h", LINUX is defined, so that in user code, the compiler knows when to select the Linux related code.

Software Application Development Notes

Currently, the XOSL APU PSP only supports co_streams for hardware/software communications. The HW_STREAM_READ, HW_STREAM_WRITE and HW_STREAM_CLOSE primitives are recommended in software-side coding for higher efficiency and thus faster execution time.

The user C code is being built using cross compiler Denx EDLK toolchain "ppc_4xx-gcc". If your code previously runs in standalone mode, and you want to adapt it to run on Linux, you need to eliminate all the Xilinx-related functions and header files, and replace them with Linux ones. Also, since your application runs in user mode, don't use any privilege mode asm instructions, such as "mtmsr".

XOSL Development Environment Setup

Building applications for Xilinx embedded Linux requires a Linux development environment. You can either use a dedicated Linux computer for this purpose, or use a virtual machine. For this project, I chose to download CentOS 5.2 as my Linux OS, running it on a VMware virtual machine. Any Linux 2.6 system will work.

"git" is needed for installing XOSL. You can install it as follows:

```
su -c 'rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm'  
su -c 'yum -y install git'
```

You may need to install "ncurses" for using menuconfig.

```
yum -y install ncurses-devel
```

The XOSL information is available online at <http://xilinx.wikidot.com/open-source-linux>.

To clone the Xilinx Linux Kernel tree run:

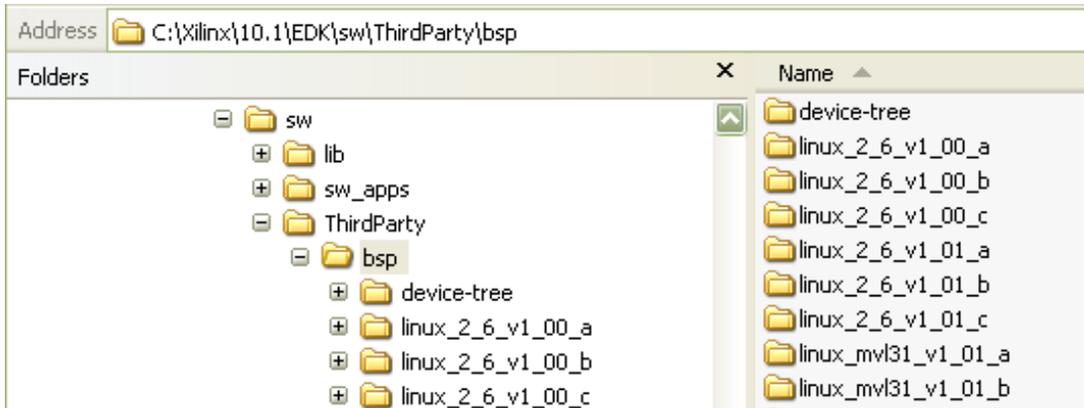
```
git clone git://git.xilinx.com/linux-2.6-xlnx.git
```

Also, clone the device tree bsp using git:

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

Copy the device-tree to the following directory on your PC:

```
[Xilinx Installation Directory]\10.1\EDK\sw\ThirdParty\bsp
```



The EDK runs on Windows, so we need to mount a windows share folder on the Linux machine for sharing documents between your Linux machine and windows PC.

```
mount -t cifs //[IP address of the Windows host]/[folder name] [/dir in Linux machine] -o username=[xxx],password=[***]
```

You need to download a cross compiler for PowerPC, such as DENX_ELDK. You can download ISO image ppc-2008-04-01_amcc.iso using FTP from site ftp.sunet.se at /pub/Linux/distributions/eldk/4.2/ppc-linux-x86/iso/.

To mount ISO:

```
su
mkdir /mnt/iso
mount myiso.iso /mnt/iso/ -t iso9660 -o ro,loop=/dev/loop0
```

To install:

```
cd /mnt/iso
./install -d /home/meixu/Xilinx/DENX_ELDK ppc_4xx
```

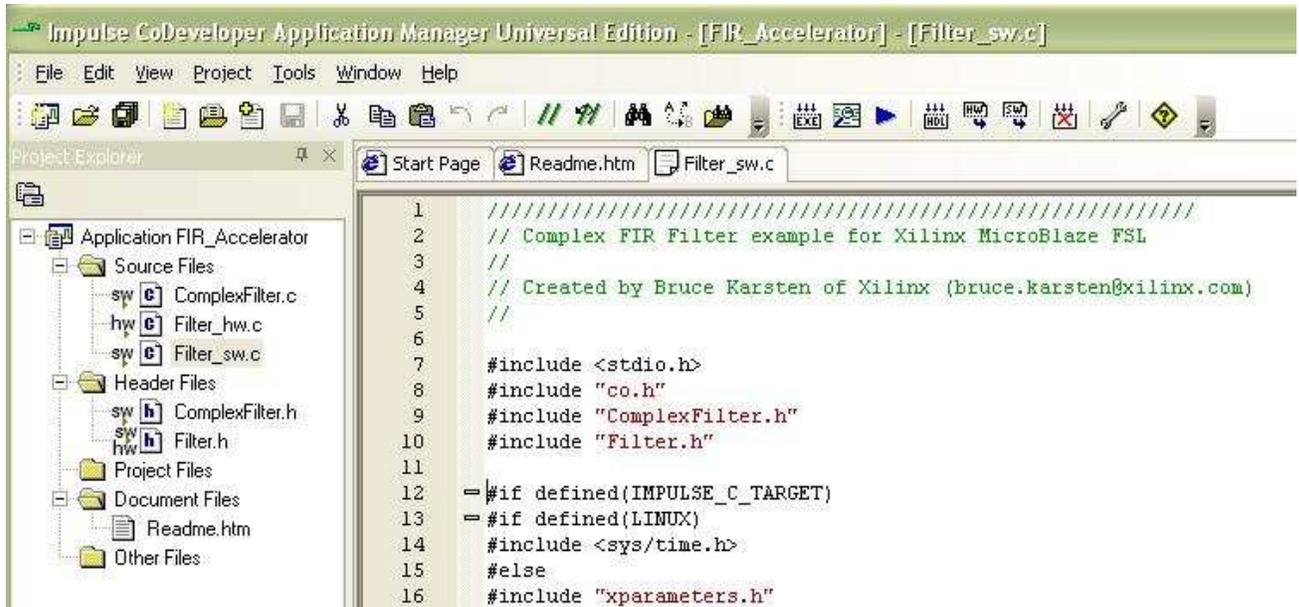
To set path and environment variable:

```
PATH=/home/meixu/Xilinx/DENX_ELDK/usr/bin:/home/meixu/Xilinx/DENX_ELDK/bin:
$PATH --let the ELDK/usr/bin dir first to use the new binutils.
```

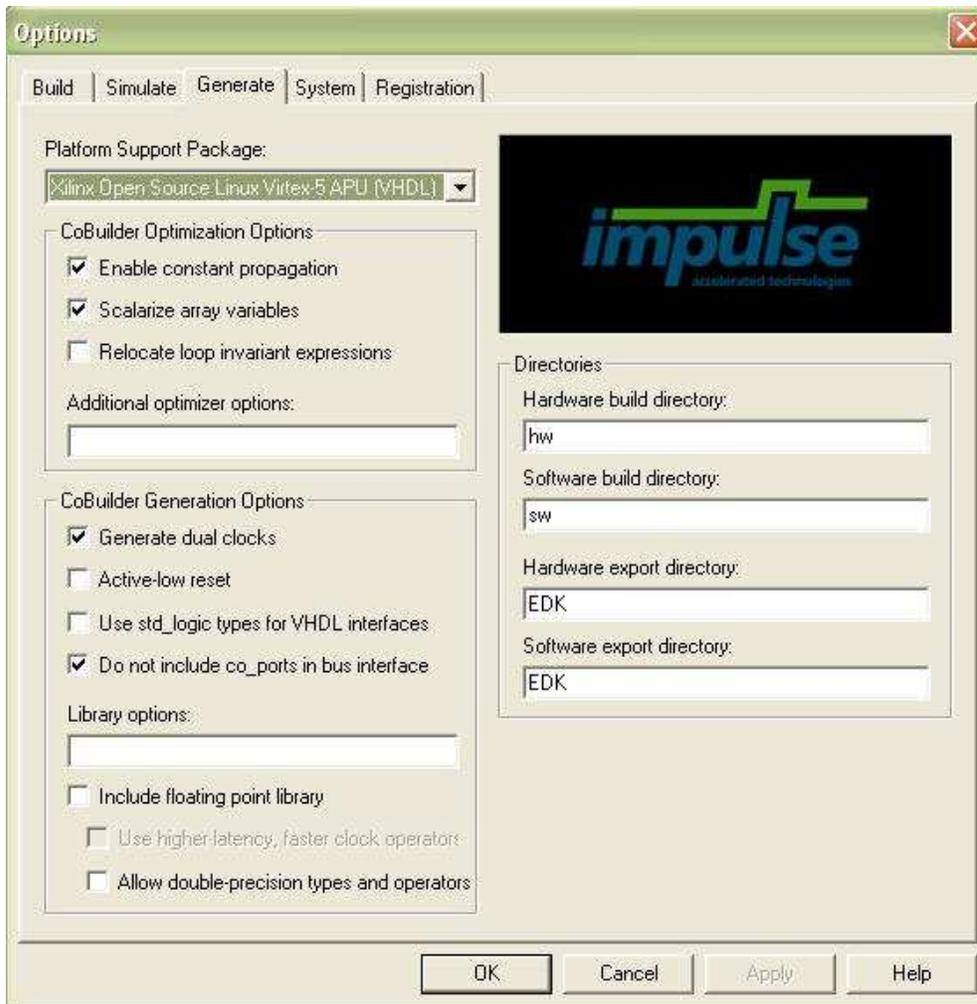
```
export CROSS_COMPILE=ppc_4xx-
```

Using CoDeveloper to Generate Hardware and Software

Copy the ComplexFIR project folder "ComplexFIR_XOSL_PPC440" to your windows share directory. Open the project in CoDeveloper Application Manager.



Open the Project → Options Dialogue, and choose "Xilinx Open Source Linux Virtex-5 APU (VHDL)" as the Platform Support Package. Checking the "Generate dual clock" will allow the Impulse hardware module to use a different clock other than the APU clock of PowerPC.



Next, generate HDL code by clicking the "HDL" button below:



Export the generated hardware to designated directory "EDK" by clicking the "HW" button.



Export the generated software by clicking the "SW" button.



The directory structure is shown below. The HDL code and drivers are in the "EDK" directory. Software code is in the "user_app" directory.



In next step, we use Xilinx Platform Studio to build a PowerPC system with the generated Impulse module.

Create a new XPS project in the "EDK" directory above. In Base System Builder (BSB), select Xilinx ML507 as the development platform.



Select PowerPC as the processor.

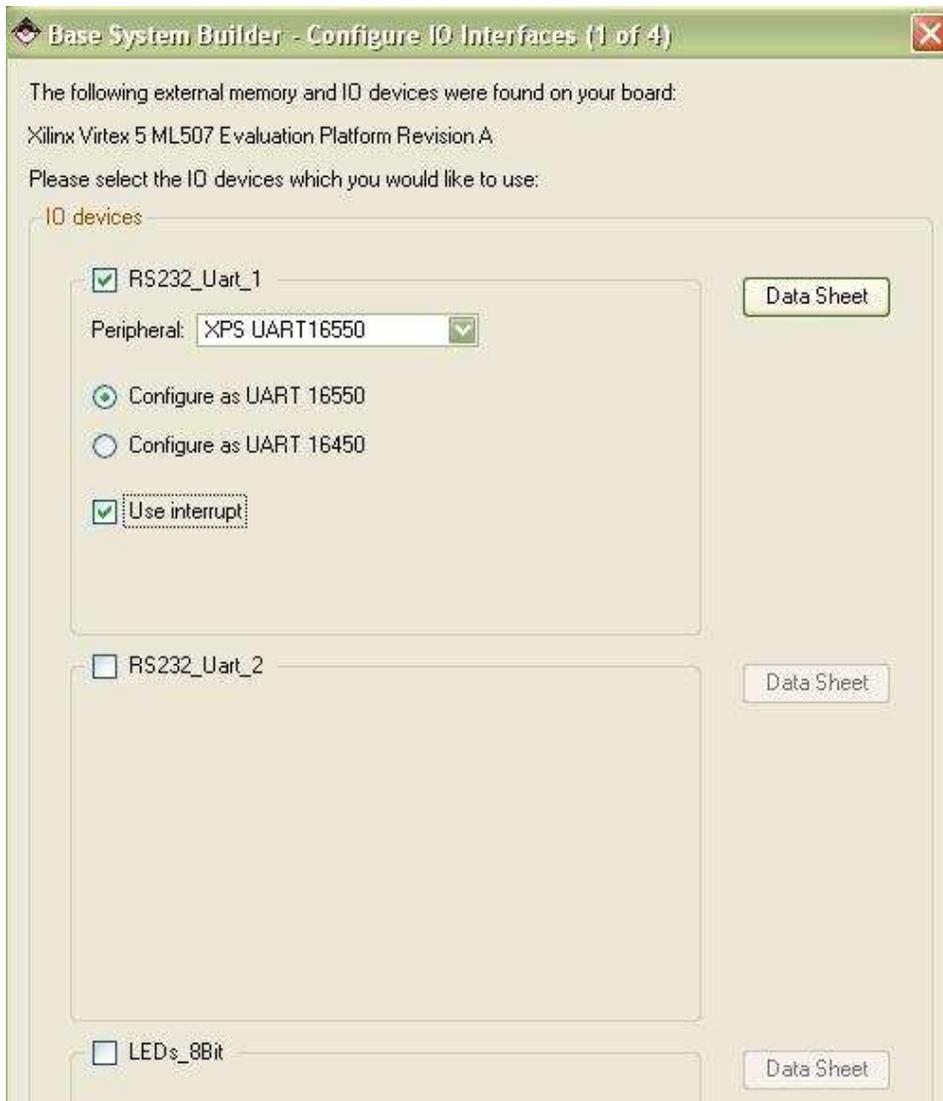


Next, choose processor clock to be 400 MHz, and bus clock 100 MHz. Enable cache, and disable the FPU.



In IO configuration, select device RS232_Uart_1 with type XPS UART 16550, and check "Use interrupt".

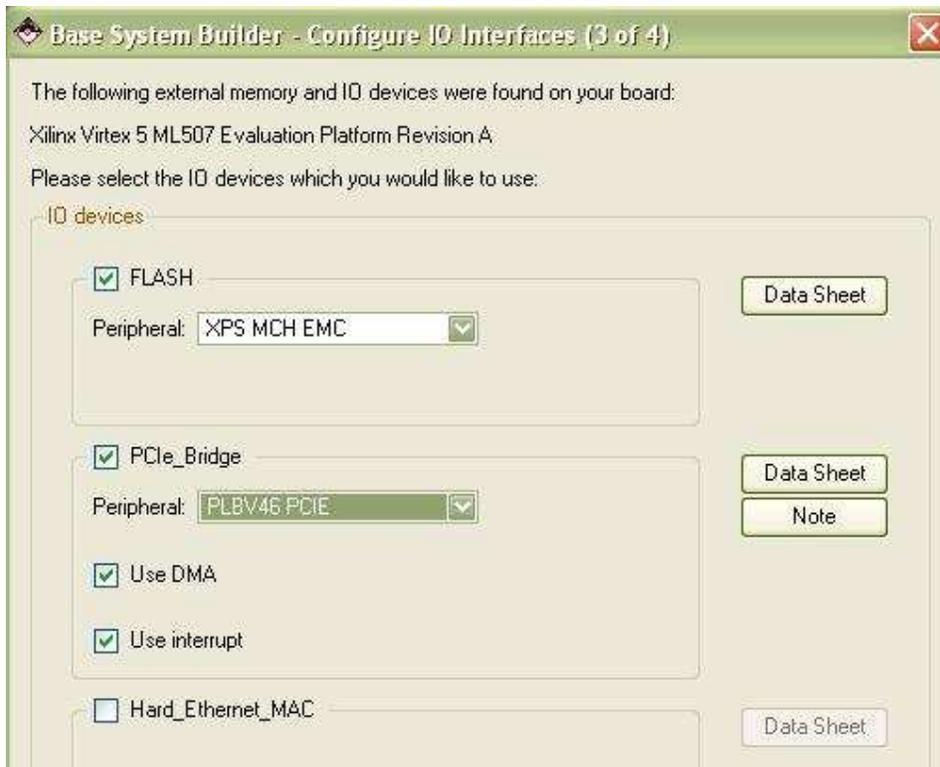
Unselect the RS232_Uart_2 and the LEDs_8Bit.



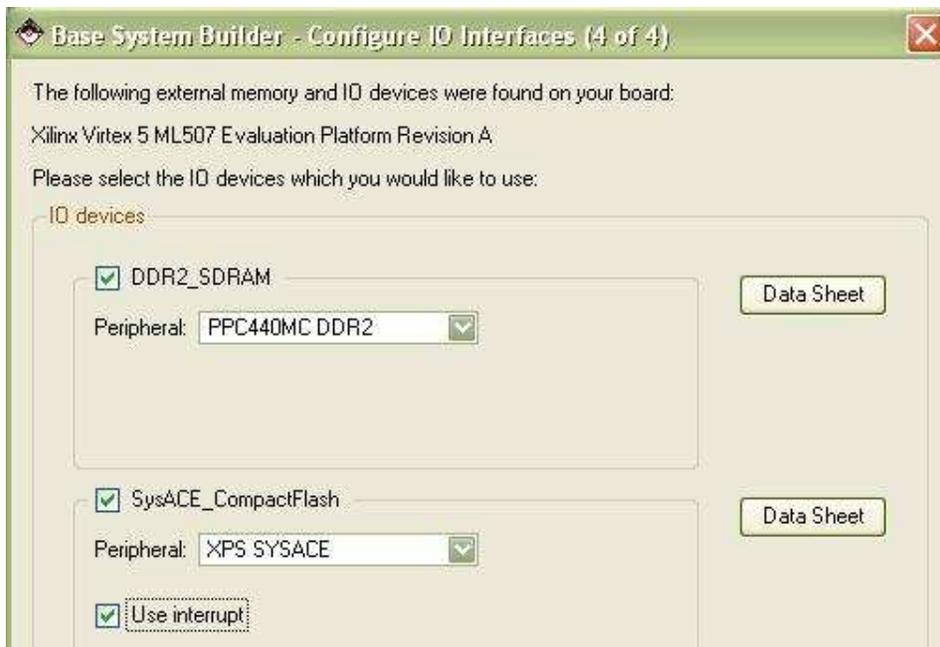
On the next page, select the IIC_EEPROM only.



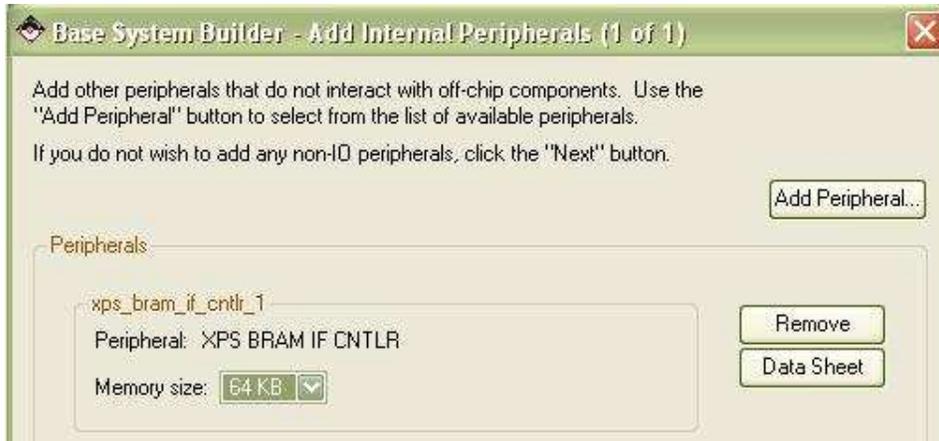
Next, select the FLASH and the PCIe_Bridge devices.



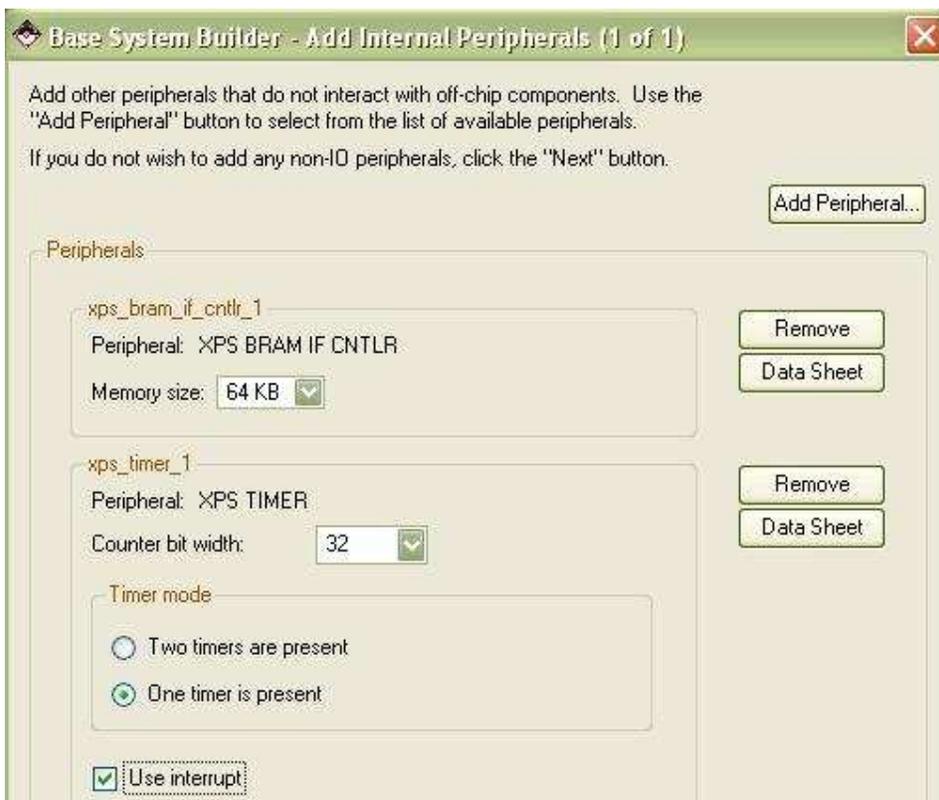
Select the DDR2_SDRAM and the SysACE_CompactFlash.



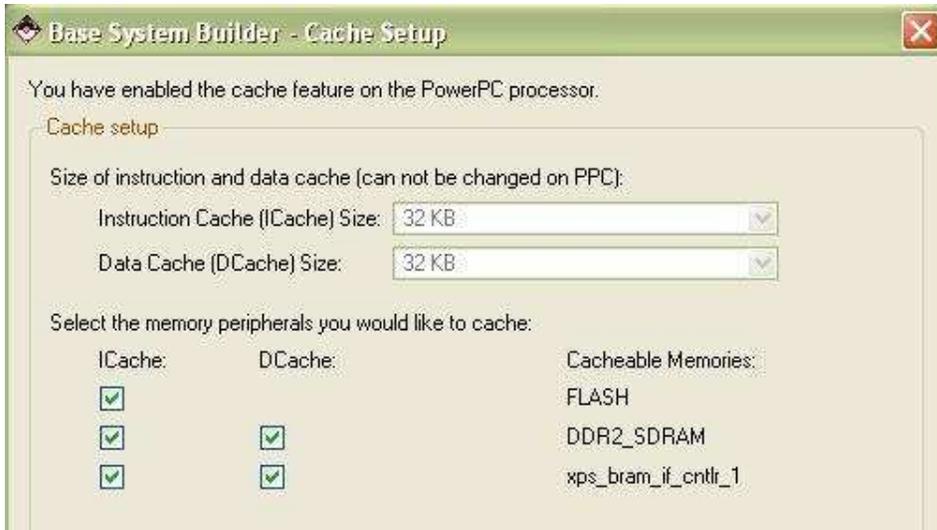
Next, enlarge the memory size of XPS BRAMs to 64 KB.



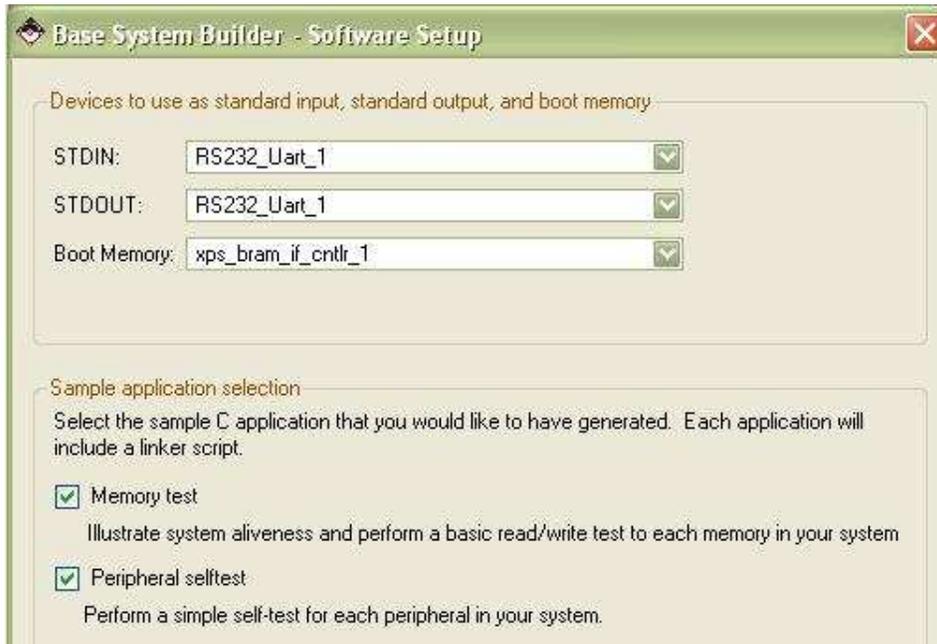
Add a peripheral "xps_timer_1" to the system. Choose one timer mode, and use interrupt.



Next, select the cache options.



On the next page, select the sample applications as you wish.



Next, click OK to accept memory settings for the sample applications. A page of design summary appears. Click "generate" to build the system.

Base System Builder - System Created

Below is a summary of the system you have created. Please review the information below. If it is correct, hit <Generate> to enter the information into the XPS data base and generate the system files. Otherwise return to the previous page to make corrections.

Processor: ppc440_0
 Processor clock frequency: 400.00 MHz
 Bus clock frequency: 100.00 MHz
 On Chip Memory : 64 KB
 Total Off Chip Memory : 288 MB
 - FLASH = 32 MB

The address maps below have been automatically assigned. You can modify them using the editing features of XPS.

PLB Bus : PLB_V46 Inst. name: plb_v46_0 Attached Components:

Core Name	Instance Name	Base Addr	High Addr
xps_bram_if_cntrl	xps_bram_if_cntrl_1	0xFFFF0000	0xFFFFFFFF
xps_uart16550	RS232_Uart_1	0x83E00000	0x83E0FFFF
xps_iic	IIC_EEPROM	0x81600000	0x8160FFFF
xps_mch_emc	FLASH	0xD8000000	0xD9FFFFFF
plbv46_pcie	PCIe_Bridge	0x85C00000	0x85C0FFFF
plbv46_pcie	PCIe_Bridge_C_IPIFBA	0xA0000000	0xBFFFFFFF
xps_sysace	SysACE_CompactFlash	0x83600000	0x8360FFFF
xps_timer	xps_timer_1	0x83C00000	0x83C0FFFF
xps_central_dma	xps_central_dma_0	0x80200000	0x8020FFFF
xps_intc	xps_intc_0	0x81800000	0x8180FFFF

PPC440MC Bus : ppc440_0_PPC440MC Attached Components:

Core Name	Instance Name	Base Addr	High Addr
ppc440mc_ddr2	DDR2_SDRAM	0x00000000	0x0fffffff

Here is the bus interface view of the PowerPC system you just built:

Xilinx Platform Studio - C:\VM_SharedFolder\ComplexFIR_XDSL_PPC440\EDK\system.xmp - [System Assembly View1]

File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help

Project Information Area

Project Applications IP Catalog

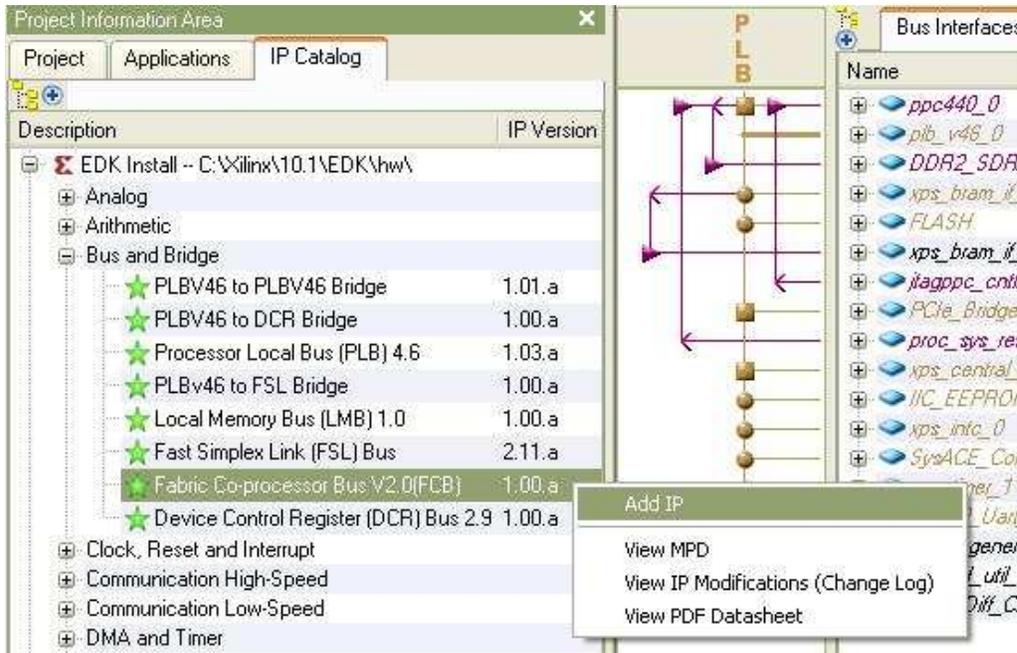
Software Projects

- Add Software Application Project...
- Default: ppc440_0_bootloop
- Project: TestApp_Memory
 - Processor: ppc440_0
 - Executable: C:\VM_SharedFolder\ComplexFIR_XDSL_PPC
 - Compiler Options
 - Sources
 - Headers
- Project: TestApp_Peripheral
 - Processor: ppc440_0
 - Executable: C:\VM_SharedFolder\ComplexFIR_XDSL_PPC
 - Compiler Options
 - Sources
 - Headers

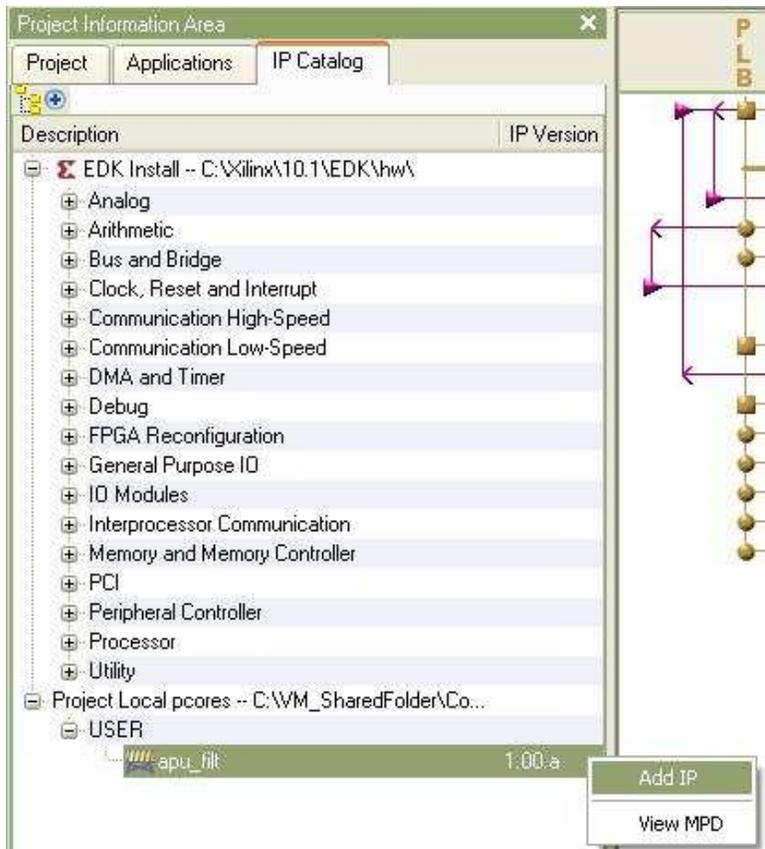
Bus Interfaces Ports Addresses

Name	Bus Connection	IP Type	IP Version
ppc440_0		ppc440_virtex5	1.01.a
plb_v46_0		plb_v46	1.03.a
DDR2_SDRAM		ppc440mc_ddr2	2.00.a
xps_bram_if_cntrl_1		xps_bram_if_cntrl	1.00.a
FLASH		xps_mch_emc	2.00.a
xps_bram_if_cntrl_1_bram		bram_block	1.00.a
itagppc_cntrl_0		itagppc_cntrl	2.01.c
PCIe_Bridge		plbv46_pcie	3.00.a
proc_sys_reset_0		proc_sys_reset	2.00.a
xps_central_dma_0		xps_central_dma	2.00.b
IIC_EEPROM		xps_iic	2.00.a
xps_intc_0		xps_intc	1.00.a
SysACE_CompactFlash		xps_sysace	1.00.a
xps_timer_1		xps_timer	1.00.a
RS232_Uart_1		xps_uart16550	2.00.b
clock_generator_0		clock_generator	2.01.a
FLASH_util_bus_split_0		util_bus_split	1.00.a
PCIe_Drv_Clk		util_ds_buf	1.00.a

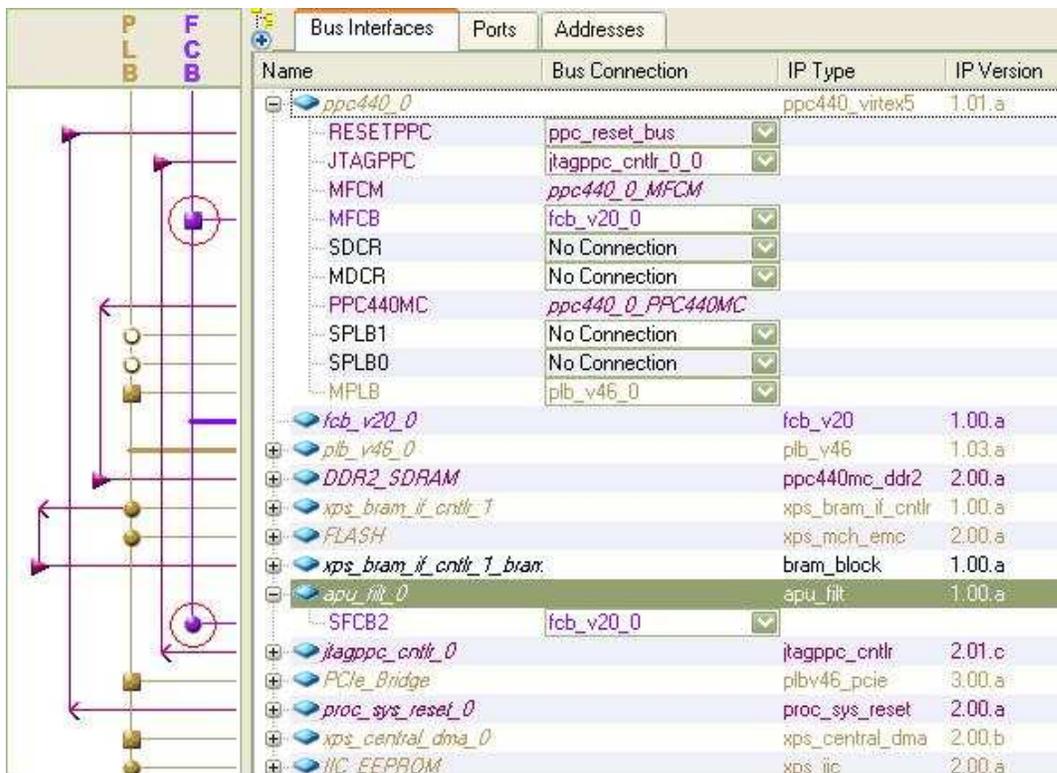
In order to use the APU feature, we need to add a Fabric Co-processor Bus to the system.



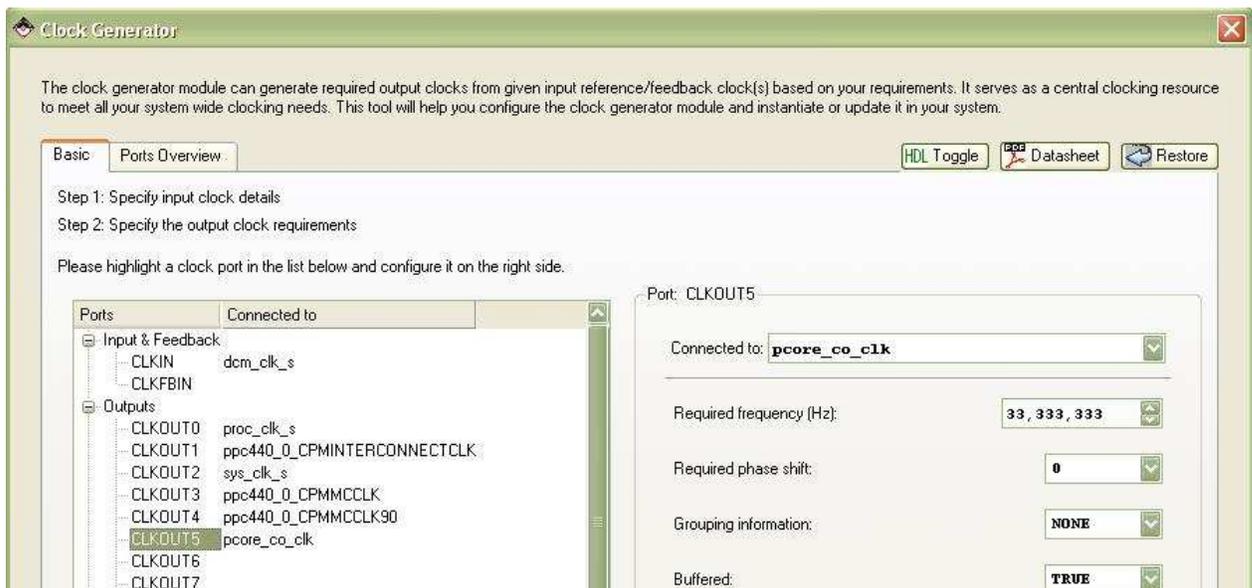
Next, add the Impulse generated module **apu_filt** from the Project Local pcores\USER.



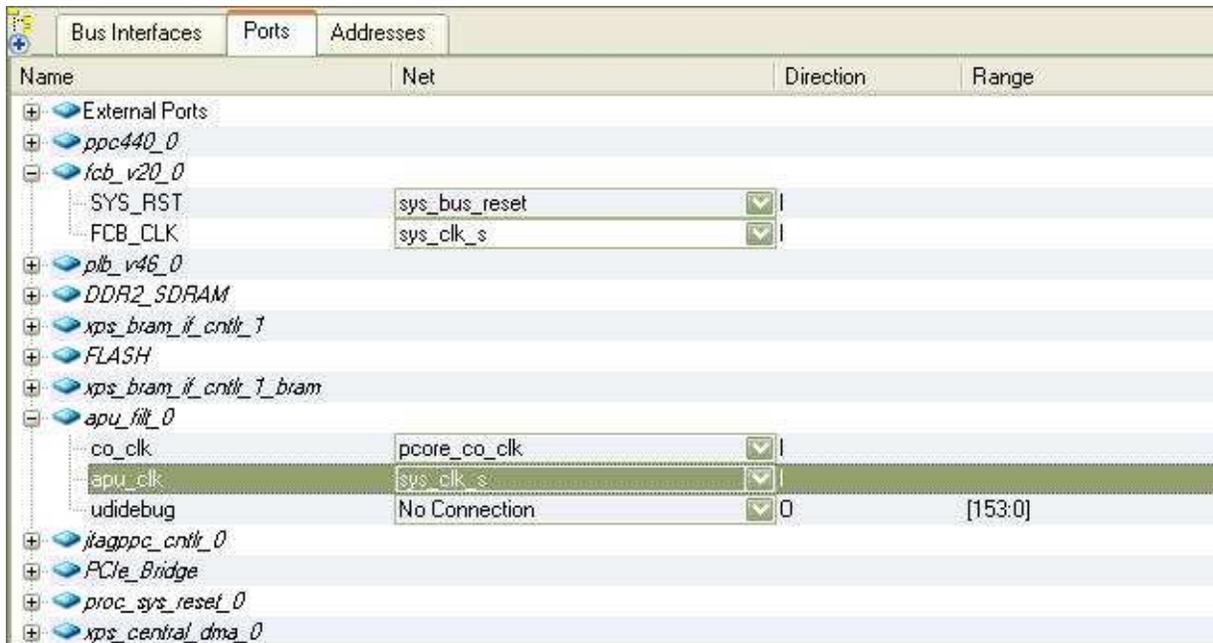
Next, connect the FCB bus to both ***ppc440_0*** and ***apu_filt_0*** as shown below:



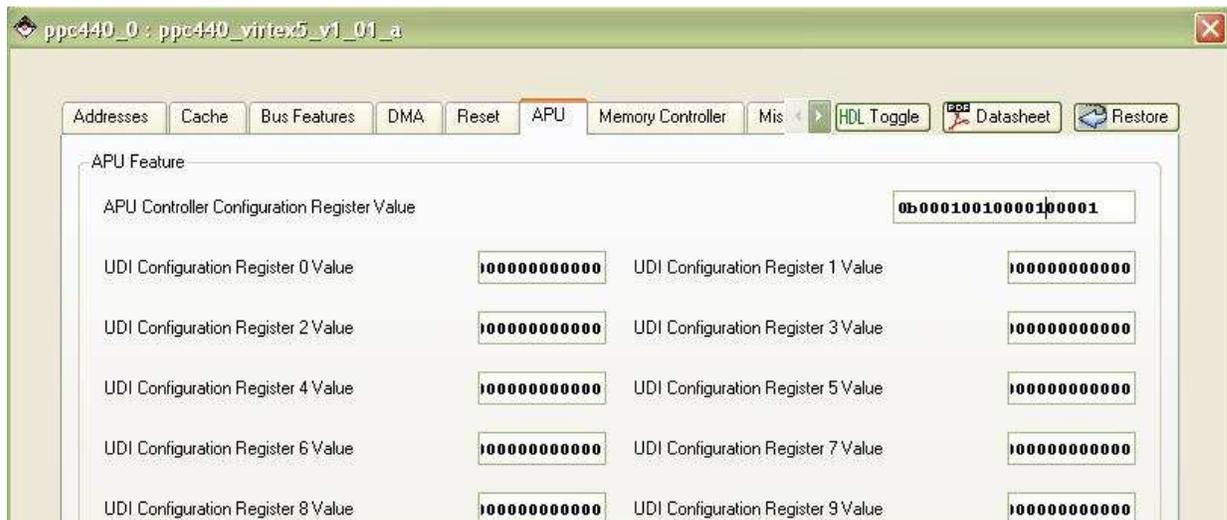
Open the "Clock Generator" window by double-clicking the module **clock_genrator_0**. Add another clock output "pcore_co_clk" of frequency 33.333333 MHz as shown below.



In the "Ports" tab, connect the reset and clock ports of **fcb_v20_0**. And connect the co_clk and apu_clk ports of **apu_filt_0** as shown below.



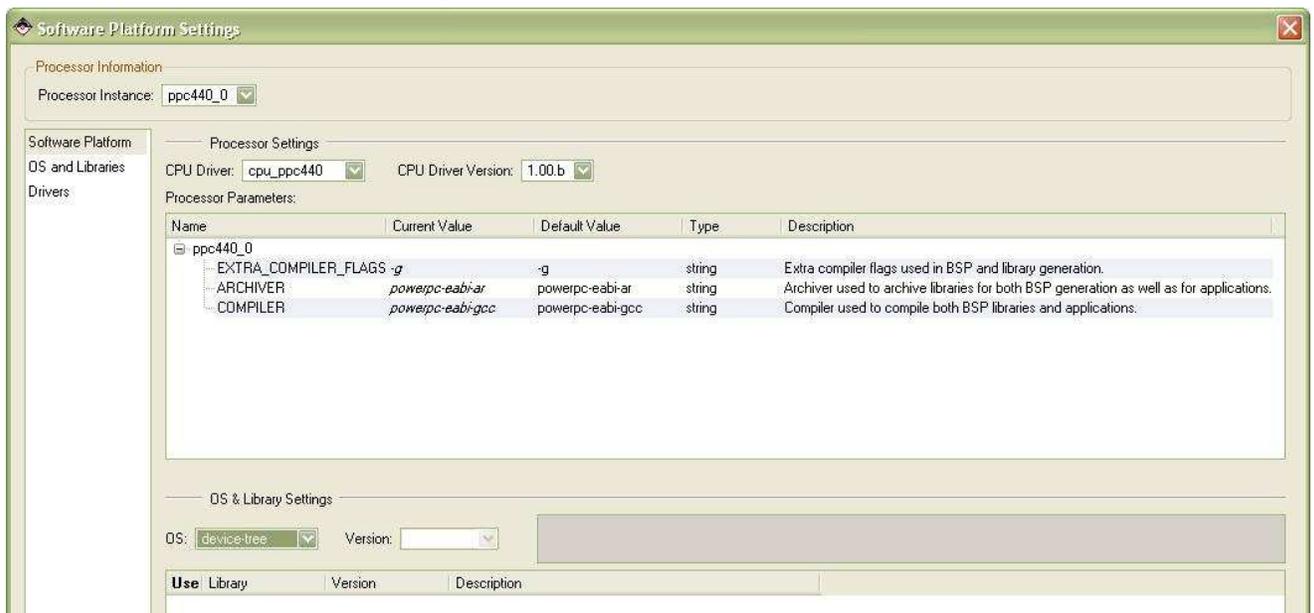
Double-click the **ppc440_0** to open the dialogue below. In the APU tab, set the APU Controller Configuration Register Value to 0b0010010000100001.



Next, go to the Addresses tab, and generate addresses for the system.

Instance	Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection
plb_v46_0	C_BASEADDR			U	Not Applicable	
PCIe_Bridge	C_BASEADDR	0x85c00000	0x85c0ffff	64K	SPLB	plb_v46_0
xps_bram_if_cntlr_1	C_BASEADDR	0xffff0000	0xfffffff	64K	SPLB	plb_v46_0
xps_central_dma_0	C_BASEADDR	0x80200000	0x8020ffff	64K	SPLB	plb_v46_0
IIC_EEPROM	C_BASEADDR	0x81600000	0x8160ffff	64K	SPLB	plb_v46_0
xps_intc_0	C_BASEADDR	0x81800000	0x8180ffff	64K	SPLB	plb_v46_0
SysACE_CompactFlash	C_BASEADDR	0x83600000	0x8360ffff	64K	SPLB	plb_v46_0
xps_timer_1	C_BASEADDR	0x83c00000	0x83c0ffff	64K	SPLB	plb_v46_0
RS232_Uart_1	C_BASEADDR	0x83e00000	0x83e0ffff	64K	SPLB	plb_v46_0
ppc440_0	C_IDCR_BASEADDR	0b0000000000	0b0011111111	256	Not Connected	
PCIe_Bridge	C_IPFBAR_0	0xa0000000	0xbfffffff	512M	SPLB	plb_v46_0
DDR2_SDRAM	C_MEM_BASEADDR	0x00000000	0x0fffffff	256M	PPC440MC	ppc440_0_PPC440MC
FLASH	C_MEM0_BASEADDR	0x94000000	0x95ffffff	32M	SPLB	plb_v46_0
ppc440_0	C_SPLB0_RNG_MC_BASEADDR			U	Not Connected	
ppc440_0	C_SPLB1_RNG_MC_BASEADDR			U	Not Connected	

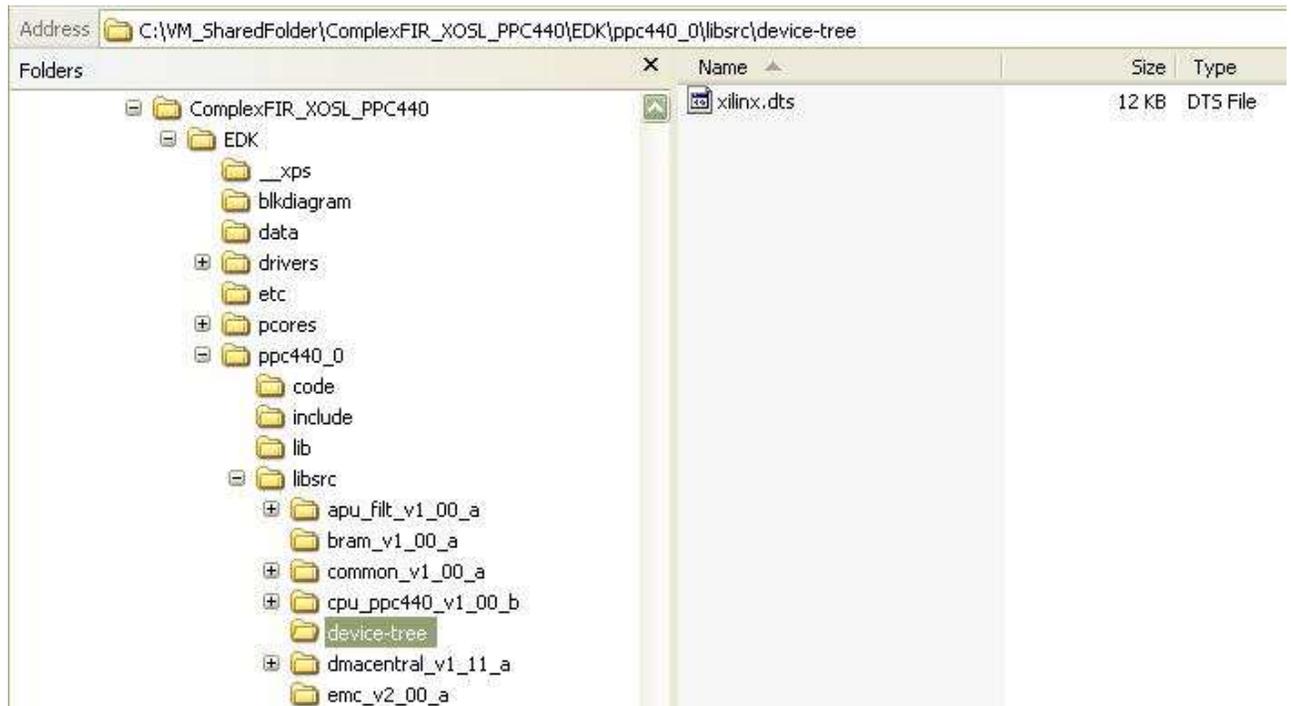
Open the Software Platform Settings Dialogue. Select "device-tree" as the OS.



In the OS and Libraries view, type "RS232_Uart_1" as the console device, and "console=ttyS0 root=/dev/ram rw ip=off" as the bootargs.



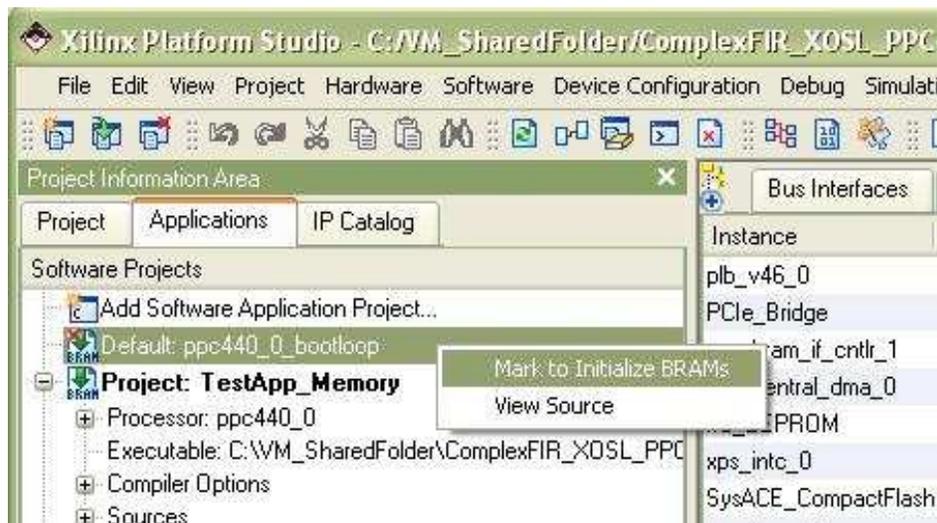
Next, click Software → Generate Libraries and BSPs. This will create a file “Xilinx.dts” in directory EDK\ppc440_0\libsrc\device-tree.



Rename the DTS file to virtex440-m1507.dts, and copy it to linux-2.6-xlnx/arch/powerpc/boot/dts/

Go back to XPS, Hardware → Generate Bitstream. This will take 10 minutes or more to finish, depending on your PC speed.

In the Applications Tab, select the **ppc440_0_bootloop** as the application for initializing BRAMs.



Building the XOSL OS and File System on the Linux Machine

Go to the XOSL main directory: linux-2.6-xlnx

```
make ARCH=powerpc 44x/virtex5_defconfig  
make ARCH=powerpc menuconfig
```

Make changes to the following options:

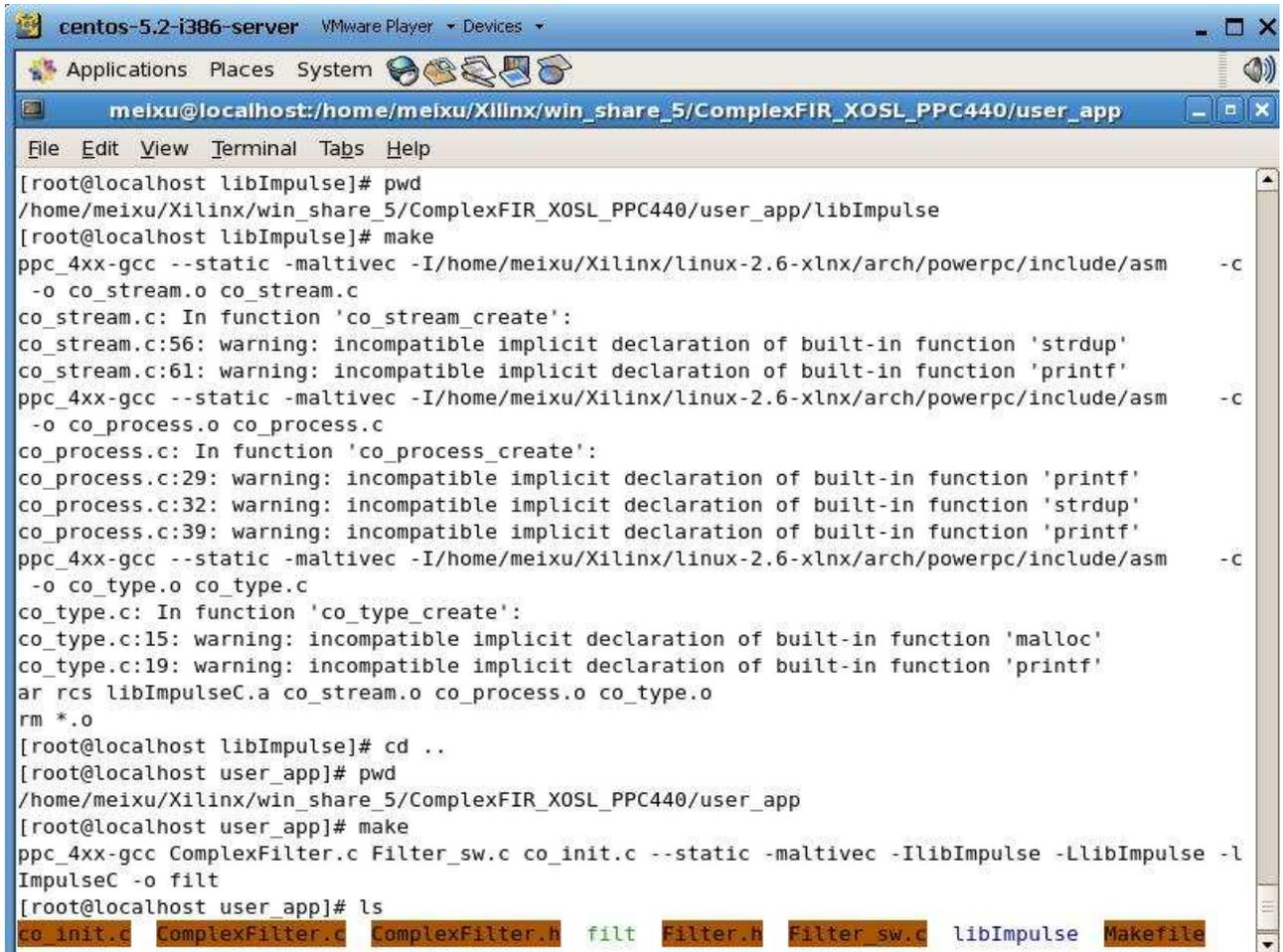
- Kernel options → Initial kernel command string: console=ttyS0
boot=/dev/ram rw ip=off
- File systems → Ext3 journalling file system support
- Device Drivers → Block devices → Xilinx systemACE support
- Exit and save changes.

Next, download "ramdisk.image.gz" from <http://xilinx.wikidot.com/open-source-linux>. At the end of the page, click "files", and a list of downloadable files will show. Copy the file to your windows share folder "win_share_5".

Switch to the user_app directory:

```
win_share-5/ComplexFIR_XOSL_PPC440/user_app
```

User application source files and a Makefile are in the user_app folder. The Impulse library source files and a Makefile are in the libImpulse subdirectory. Use command "make" to build the Impulse library "libImpulse.a", and in the user_app directory, use command "make" to build the executable file "filt".

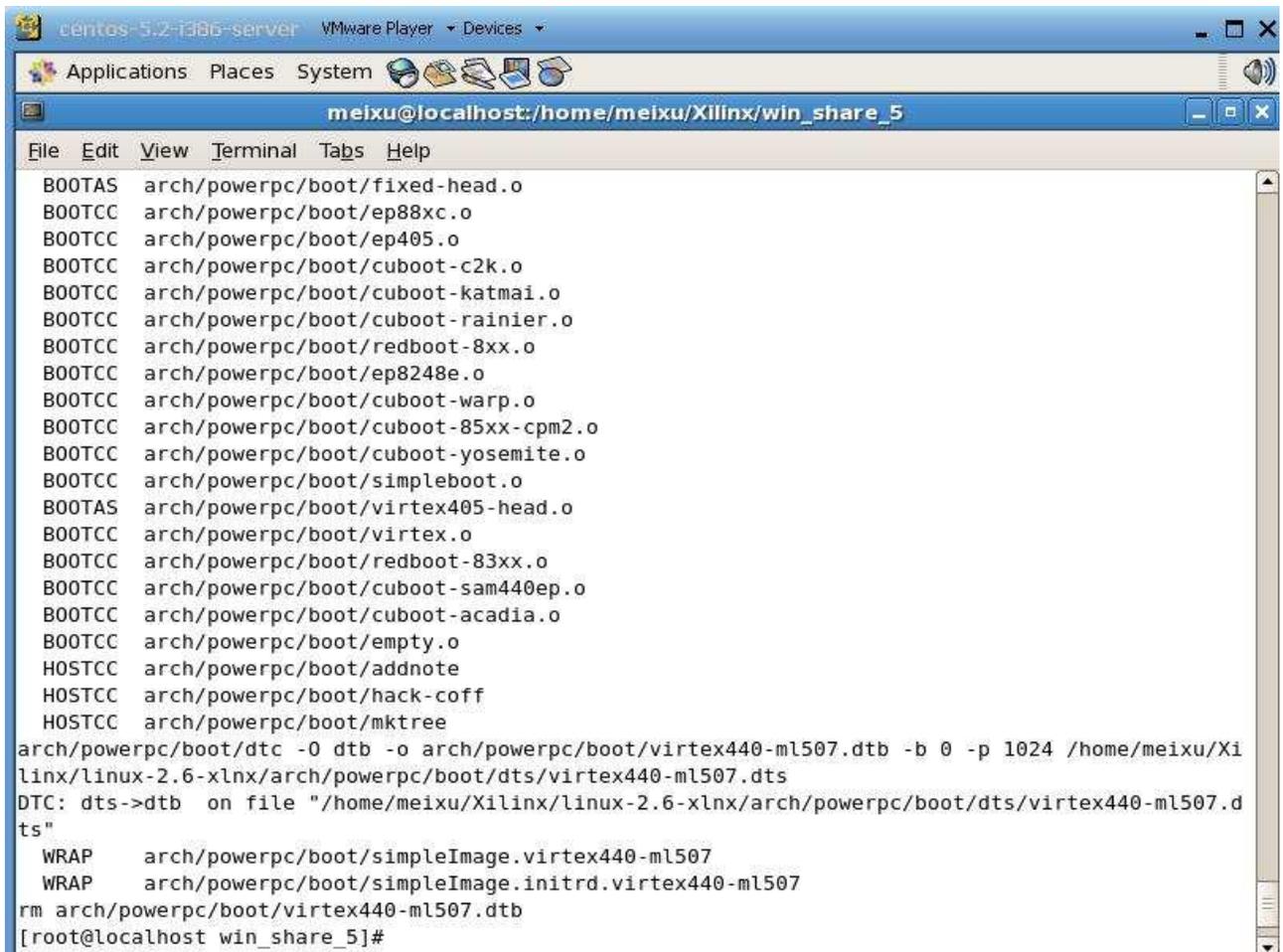


```
centos-5.2-i386-server VMware Player Devices
Applications Places System
meixu@localhost:/home/meixu/Xilinx/win_share_5/ComplexFIR_XOSL_PPC440/user_app
File Edit View Terminal Tabs Help
[root@localhost libImpulse]# pwd
/home/meixu/Xilinx/win_share_5/ComplexFIR_XOSL_PPC440/user_app/libImpulse
[root@localhost libImpulse]# make
ppc_4xx-gcc --static -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -c
-o co_stream.o co_stream.c
co_stream.c: In function 'co_stream_create':
co_stream.c:56: warning: incompatible implicit declaration of built-in function 'strdup'
co_stream.c:61: warning: incompatible implicit declaration of built-in function 'printf'
ppc_4xx-gcc --static -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -c
-o co_process.o co_process.c
co_process.c: In function 'co_process_create':
co_process.c:29: warning: incompatible implicit declaration of built-in function 'printf'
co_process.c:32: warning: incompatible implicit declaration of built-in function 'strdup'
co_process.c:39: warning: incompatible implicit declaration of built-in function 'printf'
ppc_4xx-gcc --static -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -c
-o co_type.o co_type.c
co_type.c: In function 'co_type_create':
co_type.c:15: warning: incompatible implicit declaration of built-in function 'malloc'
co_type.c:19: warning: incompatible implicit declaration of built-in function 'printf'
ar rcs libImpulseC.a co_stream.o co_process.o co_type.o
rm *.o
[root@localhost libImpulse]# cd ..
[root@localhost user_app]# pwd
/home/meixu/Xilinx/win_share_5/ComplexFIR_XOSL_PPC440/user_app
[root@localhost user_app]# make
ppc_4xx-gcc ComplexFilter.c Filter_sw.c co_init.c --static -maltivec -llibImpulse -llibImpulse -l
ImpulseC -o filt
[root@localhost user_app]# ls
co_init.c ComplexFilter.c ComplexFilter.h filt Filter.h Filter_sw.c libImpulse Makefile
```

For the purpose of easy operation, I create a shell script "makeDiskComplexFIR.sh" to add the ComplexFIR software application into the ramdisk and then build the XOSL system.

```
gunzip ramdisk.image.gz
mount -o loop ramdisk.image temp
rm temp/tmp/*
cp ComplexFIR_XOSL_PPC440/user_app/filt temp/tmp/
umount ramdisk.image
gzip ramdisk.image
cp ramdisk.image.gz ../linux-2.6-xlnx/arch/powerpc/boot/
cd ../linux-2.6-xlnx
make ARCH=powerpc clean
make ARCH=powerpc zImage
cp arch/powerpc/boot/simpleImage.initrd.virtex440-
ml507.elf ../win_share_5/ComplexFIR_XOSL_PPC440
```

This will take a few minutes to finish execution. As a result, an ELF file "simpleImage.initrd.virtex440-ml507.elf" will be copied to the ComplexFIR_XOSL_PPC440 project directory.



```
centos-5.2-1386-server VMware Player Devices
Applications Places System
meixu@localhost:/home/meixu/Xilinx/win_share_5
File Edit View Terminal Tabs Help
BOOTAS arch/powerpc/boot/fixed-head.o
BOOTCC arch/powerpc/boot/ep88xc.o
BOOTCC arch/powerpc/boot/ep405.o
BOOTCC arch/powerpc/boot/cuboot-c2k.o
BOOTCC arch/powerpc/boot/cuboot-katmai.o
BOOTCC arch/powerpc/boot/cuboot-rainier.o
BOOTCC arch/powerpc/boot/redboot-8xx.o
BOOTCC arch/powerpc/boot/ep8248e.o
BOOTCC arch/powerpc/boot/cuboot-warp.o
BOOTCC arch/powerpc/boot/cuboot-85xx-cpm2.o
BOOTCC arch/powerpc/boot/cuboot-yosemite.o
BOOTCC arch/powerpc/boot/simpleboot.o
BOOTAS arch/powerpc/boot/virtex405-head.o
BOOTCC arch/powerpc/boot/virtex.o
BOOTCC arch/powerpc/boot/redboot-83xx.o
BOOTCC arch/powerpc/boot/cuboot-sam440ep.o
BOOTCC arch/powerpc/boot/cuboot-acadia.o
BOOTCC arch/powerpc/boot/empty.o
HOSTCC arch/powerpc/boot/addnote
HOSTCC arch/powerpc/boot/hack-coff
HOSTCC arch/powerpc/boot/mktree
arch/powerpc/boot/dtc -O dtb -o arch/powerpc/boot/virtex440-ml507.dtb -b 0 -p 1024 /home/meixu/Xi
linux/linux-2.6-xlnx/arch/powerpc/boot/dts/virtex440-ml507.dts
DTC: dts->dtb on file "/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/boot/dts/virtex440-ml507.d
ts"
WRAP arch/powerpc/boot/simpleImage.virtex440-ml507
WRAP arch/powerpc/boot/simpleImage.initrd.virtex440-ml507
rm arch/powerpc/boot/virtex440-ml507.dtb
[root@localhost win_share_5]#
```

Downloading and Execution

Now, the bitstream and the ELF file are ready, and we can download them to the FPGA, and run Linux.

- Connect the ML507 board with Xilinx USB Platform Cable via JTAG, and also connect the RS-232 serial cable and power adaptor. Turn on the power switch.
- Open the Tera Term application, select type "Serial", and make sure the settings are 9600-8-N-1.
- In Xilinx XPS, Device Configuration → Download Bitstream. This will program the Virtex-5 FPGA via JTAG connection.
- Debug → Launch XMD. For the first time open XMD, the XMD Debug Options Dialogue will appear. Just click OK to accept the default settings.
- Download the generated ELF file to FPGA and let it run as shown below.

```
D:\Xilinx\10.1\EDK\bin\nt\bash.exe
I-Cache (Data).....0x70000000 - 0x70007fff
I-Cache (TAG).....0x70008000 - 0x7000ffff
D-Cache (Data).....0x78000000 - 0x78007fff
D-Cache (TAG).....0x78008000 - 0x7800ffff
DCR.....0x78020000 - 0x78020fff
TLB.....0x70020000 - 0x70023fff

Connected to "ppc" target. id = 0
Starting GDB server for "ppc" target (id = 0) at TCP port no 1234
XMD% dow ../simpleImage.initr.d.virtex440-m1507.elf
System Reset .... DONE
Downloading Program -- ../simpleImage.initr.d.virtex440-m1507.elf
section, .text: 0x00400000-0x00408d03
section, .data: 0x00409000-0x0040abdf
section, __builtin_cmdline: 0x0040abe0-0x0040addf
section, .kernel:dtb: 0x0040ade0-0x0040e104
section, .kernel:ulinux.strip: 0x0040f000-0x005a843a
section, .kernel:initr.d: 0x005a9000-0x0079c19d
section, .bss: 0x0079d000-0x007a9ddb
Setting PC with Program Start Address 0x004000b4

XMD% run
Info:Processor started. Type "stop" to stop processor

RUNNING> XMD%
```

Watch the Linux boot display in the Tera Term window:

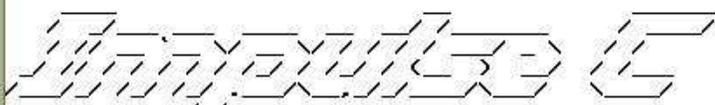
```
Tera Term - COM1 VT
File Edit Setup Control Window Help
initr.d head: 0x1f8b0808

Linux/PowerPC load: console=ttyS0 root=/dev/ram rw ip=off
Finalizing device tree... flat tree at 0x7aa300
Using Xilinx Virtex440 machine description
Linux version 2.6.28-rc6 (root@localhost.localdomain) (gcc version 4.2.2) #37 PR
EEMPT Wed Jan 21 18:28:09 PST 2009
Found initr.d at 0xc05a9000:0xc079c19e
Zone PFN ranges:
 DMA      0x00000000 -> 0x00010000
 Normal   0x00010000 -> 0x00010000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
 0: 0x00000000 -> 0x00010000
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65024
Kernel command line: console=ttyS0 root=/dev/ram rw ip=off
Xilinx intc at 0x81800000 mapped to 0xfdf000
PID hash table entries: 1024 (order: 10, 4096 bytes)
clocksource: timebase mult[0x0000] shift[22] registered
Console: colour dummy device 80x25
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 253952k/262144k available (3312k kernel code, 7912k reserved, 128k data,
137k bss, 160k init)
Calibrating delay loop... 798.72 BogoMIPS (lpj=1597440)
Mount-cache hash table entries: 512
net_namespace: 636 bytes
NET: Registered protocol family 16
PCI: Probing PCI hardware
NET: Registered protocol family 1
```

The user application executable file "filt" is located in /tmp folder in the file system.

```
Tera Term - COM1 VT
File Edit Setup Control Window Help
Device Tree Probing 'ethernet'
xilinx_lltemac 81c00000.ethernet: MAC address is now 2: 0: 0: 0: 0: 0
xilinx_lltemac 81c00000.ethernet: XLLTemac: using DMA mode.
XLLTemac: DCR address: 0x80
XLLTemac: buffer descriptor size: 32768 (0x8000)
XLLTemac: Allocating DMA descriptors with kmalloc(6>XLLTemac: (buffer_descriptor
_init) phy: 0xd0, virt: 0x0, size: 0xf890000
XTemac: PHY detected at address 7.
xilinx_lltemac 81c00000.ethernet: eth0: Xilinx IEMAC at 0x81C00000 mapped to 0xD
1016000, irq=18
mice: PS/2 mouse device common for all mice
i2c /dev entries driver
TCP cubic registered
NET: Registered protocol family 17
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RAMDISK: Compressed image found at block 0
EXT3-fs warning: feature flags set on rev 0 fs, running e2fsck is recommended
EXT2-fs warning: feature flags set on rev 0 fs, running e2fsck is recommended
EXT2-fs warning: mounting unchecked fs, running e2fsck is recommended
UFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 160k init
### Application running ...
root:~> ls
bin      etc      home    linuxrc  sbin    usr
dev      ftp     lib     proc     tmp     var
root:~> cd tmp
root:/tmp> ls
filt      xinetd.pid
root:/tmp> ./filt
```

Execute the user application. As a result, the APU accelerated with Impulse hardware runs 4.82 times faster than the software only version.

```
Tera Term - COM1 VT
File Edit Setup Control Window Help

C-toFPGA Tools
for Xilinx FPGA
Platforms
-----
Complex FIR Filter Acceleration demonstration, featuring the Xilinx
PowerPC, Virtex-5 FPGA and Impulse C-to-FPGA tools.
=====Running the Software-Only Version=====
--> Begin filtering two slots
*****
--> Done filtering two slots, execution time : 0.737921 seconds
=====Running the Accelerated Version=====
--> Begin filtering two slots
*****
--> Done filtering two slots, execution time : 0.153238 seconds
--> Acceleration factor: 4.82X
-----> Visit www.ImpulseC.com to learn more!
root:/tmp>
```

This is the end of the tutorial.

References

1. Xilinx Open Source Linux Wiki: Configuring, Building and Loading Linux After 9/1/08
<http://xilinx.wikidot.com/configuring-building-and-loading-linux-after-9-1-08>
2. PowerPC Processor Reference Guide
http://www.xilinx.com/support/documentation/user_guides/ug011.pdf
3. PowerPC 405 Processor Block Reference Guide
http://www.xilinx.com/support/documentation/user_guides/ug018.pdf
4. Embedded Processor Block in Virtex-5 FPGAs Reference Guide
http://www.xilinx.com/support/documentation/user_guides/ug200.pdf
5. Accelerated System Performance with the APU Controller and XtremeDSP Slices
http://www.xilinx.com/support/documentation/application_notes/xapp717.pdf
6. GNU Compiler Collection (GCC): Assembler Instructions with C Expression Operands <http://gcc.gnu.org/onlinedocs/gcc-3.4.3/gcc/Extended-Asm.html>
7. AltiVec Instruction Cross-Reference
http://developer.apple.com/hardwaredrivers/ve/instruction_crossref.html
8. Impulse Support Forum: APU and Linux <http://impulse-support.com/forums/index.php?showtopic=143&st=0&p=671>