**Application Note** 

# Using Impulse C With Xilinx® Open Source Linux on a Virtex<sup>™</sup>-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

Address 🛅 C:\Xilinx\10.1\EDK\sw\ThirdParty\bsp		
Folders	× Name 🔺	
😑 🦳 sw	🔽 🗀 device-tree	
🐨 🫅 lib	linux_2_6_v1_00_a	
🕀 🧰 sw_apps	🛅 linux_2_6_v1_00_b	
🖃 🚞 ThirdParty	inux_2_6_v1_00_c	
🖃 🧰 bsp	linux_2_6_v1_01_a	
🗷 🛅 device-tree	linux_2_6_v1_01_b	
🗷 🛅 linux_2_6_v1_00_a	linux_2_6_v1_01_c	
🗷 🚞 linux_2_6_v1_00_b	linux_mvl31_v1_01_a	
🗷 🛅 linux_2_6_v1_00_c	inux_mvl31_v1_01_b	

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/Xlinx/DENK\_ELDK ppc\_4xx

To set path and environment variable:

PATH=/home/meixu/Xlinx/DENX\_ELDK/usr/bin:/home/meixu/Xlinx/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.

🗝 Impulse CoDeveloper Applica	tion Man	ager Universal Edition - [FIR_Accelerator] - [Filter_sw.c]
<u>Eile E</u> dit <u>V</u> iew Project <u>T</u> ools <u>W</u> i	ndow <u>H</u> el	P
i 🖓 😅 🕼 🗋 🚇 🍄 🖬 🐰	<b>B</b>	히 건 🔢 🕫 🛤 🎲 🝅 💂 🏙 🕺 🕨 🛗 🗒 🗬 🗮 💥 🛷 💂
eroje di Explorer 🛛 📍 🗙	🛃 Start P	age 🖉 Readme.htm 🕞 Filter_sw.c
B	1	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
Application FIR_Accelerator Source Files we ComplexFilter.c hw Filter_hw.c we Filter_sw.c Header Files wh ComplexFilter.h hw Filter.h ComplexFilter.h Broject Files Cocument Files Cocument Files Cocument Files	2 3 4 5 7 8 9 10 11 12 13 14 15 16	<pre>// Complex FIR Filter example for Xilinx MicroBlaze FSL // // Created by Bruce Karsten of Xilinx (bruce.karsten@xilinx.com) // #include <stdio.h> #include "co.h" #include "co.h" #include "ComplexFilter.h" #include "Filter.h" #include "Filter.h" #if defined(IMPULSE_C_TARGET) #if defined(LINUX) #include <sys time.h=""> #else #include "xparameters.h"</sys></stdio.h></pre>

Open the Project  $\rightarrow$  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.

Platform Support Package:	impulse
Xilmx Open Source Linux Virtex 5 APU (VHDL)  CoBuilder Optimization Options  CoBuilder Constant propagation  Scalarize array variables	Accelerated to devide gins
Relocate loop invariant expressions     Additional optimizer options:	Directories Hardware build directory: hw
CoBuilder Generation Options	Software build directory:
Generate dual clocks	sw
Active-low reset	Hardware export directory:
Use std_logic types for VHDL interfaces	EDK
Do not include co_ports in bus interface	Software export directory:
Library options:	EDK
Include floating point library Use higher latency, faster clock operators Allow double-precision types and operators	

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



Export the generated hardware to designated directory <code>"EDK"</code> by clicking the <code>"HW"</code> button.

📲 Impulse CoDevelop	er Application Manager Universal Edition - [FIR_Accelerator] - [Filter_sw.c]
<u>File E</u> dit <u>V</u> iew <u>P</u> rojec	t <u>I</u> ools <u>W</u> indow <u>H</u> elp
i 🕼 🖨 🕼 🙆 i	🎦 🖬 🗼 🏷 건기// 1/ 👭 🎼 🍻 📕 🔠 🕺 🕨 🗮 🕎 🙄 🛷 🍃
Project Explorer	Export Generated Hardware (HDL)

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

🖣 Impul	e CoDevelope	r Appli	cation l	Vanager	r Univ	/ersal	Editi	<u>]</u> - no	FIR_A	celerat	- ['س	[Fili	er_	.w.c	Ĵ				
<u>Eile E</u> dit	⊻iew Project	Tools	<u>W</u> indow	Help															
🖓 🧀	🗿 🙆 😫		8 B	間ち	C .	11 91	<b>#</b>	248 6	پ 🝓	itte 🖉	-	HDL	(WH)	[SW]	漤	3	3	-	
		<b>4</b> >	<ul> <li>ອັງ ຣເ</li> </ul>	art Page	(C) R	eadme.	htm )	🗍 Filt	er_sw.c	1	¢.	51,0155	100	Expo	ort ger	herate	d soft	ware inte	rfaces

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.

A Lwould like	to create a system for the following development board	
Board vendor:	Xilinx	V
Board name:	Virtex 5 ML507 Evaluation Platform	
Board revision:	A	
Note: Visit the	vendor website for additional board support materials.	
Vendor's Websi	ite Contact Info	
Download Third	Party Board Definition Files	
🔿 I would like	to create a system for a custom board	
O I would like	to create a system for a custom board	

#### Select PowerPC as the processor.

Architecture:	Device:	Package:	Speed grade	e:
virtex5	xc5vfx70t	M1136	-1	18
Use steppin				
	£:			
		10 I		
lect the processor	you would like to use in	this design:		
lect the processor	you would like to use in	this design:		

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

iererence clock	Processor clock	Bus clock frequency:
00.00 MHz	400.00 MH:	100.00 MHz
leset polarity:	ctive LOW	
rocessor configuratio	ń	
Debug I/F	1	
FPGA JTAG		
CPU debug use	er pins only	
CPU debug and	d trace pins	
🔘 No debug		
Powo	rPC	
Cache setup		

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.

🗢 Base System Builder - Configure 10 Interfaces (1 of 4)	
The following external memory and IO devices were found on your board: Xilinx Virtex 5 ML507 Evaluation Platform Revision A Please select the IO devices which you would like to use:	
IV devices     Iv devices     Iv RS232_Uart_1   Peripheral: XPS UART16550   Iv Configure as UART 16550     Iv Configure as UART 16450     Iv Use interrupt	Data Sheet
RS232_Uart_2	Data Sheet
LEDs_8Bit	Data Sheet

On the next page, select the IIC\_EEPROM only.

	11001	Data Shee
Peripheral:	XPS IIC	
Use int	errupt	

Next, select the FLASH and the PCIe\_Bridge devices.

nx Virtex 5 ML507 Evaluation Platform Revision A	
ase select the IO devices which you would like to use: ) <mark>devices</mark>	
Peripheral: XPS MCH EMC	Data Sheet
PCIe_Bridge Peripheral: PLBV46 PCIE	Data Sheet

#### Select the DDR2\_SDRAM and the SysACE\_CompactFlash.

Data Sheet
Data Sheet

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

dd other peripherals that do not interact with off-chip comp 'Add Peripheral'' button to select from the list of available pe	onents. Use the ripherals.
f you do not wish to add any non-IO peripherals, click the "N	vext" button.
	Add Peripheral
Peripherale	
- i enprierais	
xps_bram_if_cntlr_1	Remove
xps_bram_if_cnttr_1 Peripheral: XPS BRAM IF CNTLR	Remove

Add a peripheral "xps\_timer\_1" to the system. Choose one timer mode, and use interrupt.

🗢 Base System Builder - Add Internal Peripherals (1 of 1)	
Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals. If you do not wish to add any non-IO peripherals, click the "Next" button.	Add Peripheral
Peripherals	
xps_bram_if_cntlr_1 Peripheral: XPS BRAM IF CNTLR Memory size: 64 KB	Remove Data Sheet
xps_timer_1         Peripheral: XPS TIMER         Counter bit width:         32         Timer mode         O Two timers are present         Image: O One timer is present	Remove Data Sheet
Use interrupt	

Next, select the cache options.

i have enabled th <mark>ache setup</mark>	ie cache feature on	the PowerPC proces	ssor.
ize of instruction	and data cache (car	n not be changed or	n PPC):
Instruction C	ache (ICache) Size:	32 KB	<u>×</u>
Data Cache	(DCache) Size:	32 KB	<u>×</u>
elect the memory	peripherals you wou	uld like to cache:	
ICache:	DCache:		Cacheable Memories:
			FLASH
			DDR2_SDRAM
			xps bram if onth 1

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

STDIN-	B\$222 Hatt 1
orbini.	
STDOUT:	RS232_Uart_1
Boot Memory:	xps_bram_if_cntlr_1
ample applica	tion selection
ample applica	tion selection ple C application that you would like to have generated. Each application will
ample applica Select the sam nclude a linker	t <mark>ion selection</mark> ple C application that you would like to have generated. Each application will r script.

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.

Core Name	Instance Name	Base Addr	High Addr
xps_bram_if_cntlr	xps_bram_if_cntlr_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 Com	ponents:
Core Name	Instance Name	Base Addr	High Addr
ppc440mc ddr2	DDR2 SDRAM	0x00000000	OxOfffffff

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



×

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

Project Information Area	×	Р	Bus Interface:
Project Applications IP Catalog		LB	Name
130 C		b C b	— (i) → ppc440 0
Description	IP Version		⊕ ⇒ pib v46 0
🖨 🗶 EDK Install C:\Xilinx\10.1\EDK\hw\			— 🕀 🧼 DDR2_SDR
🕀 Analog		6	— 🕀 🧼 xps_bram_il
🕀 Arithmetic		<b>•</b>	— 🕀 🥯 FLASH
😑 Bus and Bridge			— 🕀 🧼 xps_bram_il
🚽 📩 PLBV46 to PLBV46 Bridge	1.01.a	*	— 🕀 🧼 jtagppc_cnti
- 🛧 PLBV46 to DCR Bridge	1.00.a	u u u	—   🕀 🧼 PCIe_Bridge
🚽 🚽 📩 🚽 🔶 🔶	1.03.a		— 🕀 🎐 proc_sys_re
🚽 🛨 🛨 🛨 🛨 🕂 🕂 🕂 🕂 🛨 🕂	1.00.a		- the state of the second
🚽 🤺 Local Memory Bus (LMB) 1.0	1.00.a		$\square$ $\square$ $\square$ $\square$ $\square$ $\square$ $\square$
- 🛧 Fast Simplex Link (FSL) Bus	2.11.a	1 5	- E SVACE CO
🚽 😳 Fabric Co-processor Bus V2.0(FCB	) 1.00.a 🖵	Add to	iner_1
🚽 🚽 📩 📩 📩 Device Control Register (DCR) Bus	2.9 1.00.a 💻	Add IP	Van
Clock, Reset and Interrupt		View MPD	genei
Communication High-Speed		View IP Modification	ns (Change Log)
Communication Low-Speed		View PDF Datashee	et D <i>ilf_C</i>
🕀 DMA and Timer	_	1	

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:

PF	Bus Interfaces Ports	Addresses		
BB	Name	Bus Connection	IP Type	IP Version
	□		ppc440_virtex5	1.01.a
	- RESETPPC	ppc_reset_bus	2	
	- JTAGPPC	jtagppc_cntlr_0_0	2	
	MFCM	ppc440_0_MFCM		
	MFCB	fcb_v20_0		
	SDCR	No Connection	2	
	MDCR	No Connection	2	
6	- PPC440MC	ppc440_0_PPC440MC		
0	- SPLB1	No Connection		
0	- SPLB0	No Connection	2	
	- MPLB	plb_v46_0	2	
	fcb_v20_0	Medicensis in	fcb_v20	1.00.a
	— ⊕ ⇒ plb_v46_0		plb_y46	1.03,a
	— 🕀 → DDR2_SDRAM		ppc440mc_ddr2	2.00.a
K 🔷	— 🚊 🧼 xps_bram_if_cntit_1		xps_bram_if_cntlr	1.00,a
	— 🗄 🧼 FLASH		xps_mch_emc	2.00,a
	— 🕀 🧼 xps_bram_il_cntil_1_bra	B/7:	bram_block	1.00.a
	😑 🥯 apu_filt_0	2000 200	apu_filt	1.00.a
	- SFCB2	fcb_v20_0	2	
	— 🕀 🧼 įtagppc_cntlr_0	And a constraint of the	jtagppc_cntlr	2.01.c
<b>1</b>	— 🕀 🧼 PCIe_Bridge		plbv46_pcie	3.00.a
K	— 🕀 🗢 proc_sys_reset_0		proc_sys_reset	2.00.a
	— 🕀 🧼 xps_central_dma_0		xps_central_dma	2.00,b
5	- IC EEPROM		XDS IIC	2.00.a

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.

clock generato eet all your syst	r module can generate required output clocks from given inpu em wide clocking needs. This tool will help you configure the	it reference/feedback clock(s) based on your requirem clock generator module and instantiate or update it in y	ents. It serves as a central clocking reso your system.
sic Ports D	verview	HDL	Toggle 🔀 Datasheet 📿 Rest
ep 1: Specify ir ep 2: Specify tł	iput clock details ne output clock requirements		
ease highlight a Ports	a clock port in the list below and configure it on the right side. Connected to	Port: CLKOUT5	
□ Input & Fe □ CLKIN □ CLKFI	edback J dcm_clk_s 31N	Connected to: pcore_co_c1k	
CLK0		Required frequency (Hz);	33, 333, 333 🔗
CLKO	UT2 sys_ck_s UT3 ppc440_CCMMCCLK	Required phase shift:	0
CLKO CLKO	UT4 ppc44U_0_CPMMCCLK90 UT5 pcore_co_clk	Grouping information:	NONE
CLKO	UTG		

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.

Đ	Bus Interfaces	Ports	Addresse	IS .		
Na	me		N	let	Direction	Range
Đ	External Ports			5141		
Ð	⇒ррс440_0					
9	fcb_v20_0					
	SYS_RST		sj	vs_bus_reset	<b>I</b>	
	FCB_CLK		sy	/s_clk_s		
Ð	> plb_v46_0					
t)	DDR2_SDRAM	1				
ŧ	>xps_bram_il_cr	ntl <u>r_</u> 1				
Đ	→ FLASH					
Ð	>xps_bram_if_cr	ntlr_1_brai	m			
-	⇒apu_filt_0					
	- co_clk		p	core_co_clk		
	apu_clk		8	ys_clk_s		
	udidebug		N	o Connection	0	[153:0]
Ð	⇒jtagppc_cntlr_t	2				
Ð	PCIe_Bridge					
Ð	<pre>&gt;&gt; proc_sys_reset</pre>	0				
Đ	xps_central_dn	na O				

Double-click the **ppc440\_0** to open the dialogue below. In the APU tab, set the APU Controller Configuration Register Value to 0b0010010000100001.

Addresses	Cache	Bus Features	DMA	Reset	APU	Memory Controller	Mis 🔞 🚺 HDL	Toggle 🧏 🖉	atasheet
APU Feat	ure								
APU Co	ntroller Con	figuration Register	Value					05000100	10000100001
UDI Cor	nfiguration F	Register 0 Value		1000	0000000	0 UDI Configura	ition Register 1 Va	lue	100000000000
UDI Cor	nfiguration F	Register 2 Value		1000	0000000	0 UDI Configura	ition Register 3 Va	lue	100000000000
UDI Cor	nfiguration F	Register 4 Value		1000	0000000	0 UDI Configura	ition Register 5 Va	lue	100000000000
UDI Cor	nfiguration F	Register 6 Value		1000	0000000	0 UDI Configura	ition Register 7 Va	lue	100000000000

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

Bus Interfaces	Ports Addresses					Generate Addresses
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	0x85c0fff	64K	SPLB SPLB	plb_v46_0
xps_bram_if_cntlr_1	C_BASEADDR	OxfffO000	Oxfffffff	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_CompactFla	shC_BASEADDR	0x83600000	0x8360ffff	64K	SPLB	plb_v46_0
xps_timer_1	C_BASEADDR	0x83c00000	0x83c0fff	64K	SPLB	plb_v46_0
RS232_Uart_1	C_BASEADDR	0x83e00000	0x83e0fff	64K	SPLB	plb_v46_0
ррс440_0	C_IDCR_BASEADDR	06000000000	060011111111	256	Not Connected	
PCIe_Bridge	C_IPIFBAR_0	ОхаООООООО	Oxbffffff	512M	SPLB	plb_v46_0
DDR2_SDRAM	C_MEM_BASEADDR	0x00000000	OxOffffff	256M	PPC440MC	ppc440_0_PPC440MC
FLASH	C_MEM0_BASEADDR	0x94000000	0x95ffffff	32M	SPLB	plb_v46_0
ppc440_0	C_SPLB0_RNG_MC_BASEAD	DR		U	Not Connected	
ррс440_0	C_SPLB1_RNG_MC_BASEAD	DR		U	Not Connected	

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

and Libraries vers	Processor Settings CPU Driver: cpu_ppc440	CPU Driver Version:	1.00.b 💟		
	Name	Current Value	Default Value	Туре	Description
	⊜ ppc440_0				
	-EXTRA_COMPILER_FLAGS	-g	-g	string	Extra compiler flags used in BSP and library generation.
	COMPLER	powerpc-eabi-ar	powerpc-eabi-ar	string	Archiver used to archive libraries for both BSP generation as well as for applicatio

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.

Processor Informati	on				
Processor Instance	e: ppc440_0 🔛				
Software Platform OS and Libraries Drivers	Configuration for OS: device-tre	e v			
	Name	Current Value	Default Value	Туре	Description
	😑 device-tree	The rest of the re	119000000000000000000000000000000000000		Contraction of the second second
	- console device	RS232_Uart_1		string	Instance name of IP core for boot console (e.g. RS232
	bootargs	console=ttvS0 root=/dev/ram rw ip=off	console=ttyS0 root=/dev/ram string		Booting arguments

Next, click Software  $\rightarrow$  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-ml507.dts, and copy it to linux-2.6-xlnx/arch/powerpc/boot/dts/

Go back to XPS, Hardware  $\rightarrow$  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.

🔶 Xilim	e Platform St	udio - C:/Vi	A_Shared	lFolder/	Comp	lexFIR_	X03	SL_PPC
File E	dit View Projec	t Hardware	Software	Device C	onfigura	ation D	ebug	Simulati
8 🗗 🕅	🛱 ିାର ଜା	X & @	00 # 🖻	0-0 😡		) 🗄 848	33	🎭 🗄 [
Project Inf	ormation Area				×	Bu	s Inte	rfaces
Project	Applications	IP Catalog				Instance		
Software I	Projects				F	lb v46	0	
Carlos and the second s			F	PCIe_Bridge				
Contract: ppc440_0_bootloop  Contract: TestApp_Memory  Contract: TestApp_Memory  Contract: ppc440_0			Mark View	ark to Initialize BRAMs iew Source			ntlr_1 na_0	
Executable: C:\VM_SharedFolder\ComplexFIR_XOSL_PPC				PPC x	xps_intc_0 SysACE_CompactFlash			

# 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  $\rightarrow$  Ext3 journalling file system support
- Device Dirvers  $\rightarrow$  Block devices  $\rightarrow$  Xilinx systemACE support
- Exit and save changes.

Next, download "ramdisk.image.gz" from http://xilinx.wikidot.com/opensource-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	_   =   ×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp	
[root@localhost libImpulse]# pwd /home/meixu/Xilinx/win_share_5/ComplexFIR_XOSL_PPC440/user_app/libImpulse [root@localhost libImpulse]# make	-
<pre>ppc_4xx-gccstatic -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -o co_stream.o co_stream.c co stream.c: In function 'co stream create':</pre>	-c
<pre>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-gccstatic -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -o co_process.c</pre>	-c
<pre>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-gccstatic -maltivec -I/home/meixu/Xilinx/linux-2.6-xlnx/arch/powerpc/include/asm -o co type.o co type.c</pre>	-c
<pre>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</pre>	
[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.cstatic -maltivec -IlibImpulse -LlibImpuls	ie -l
ImpulseC -o filt [root@localhost user_app]# ls <mark>co_init.cComplexFilter.cComplexFilter.h</mark> _filt <mark>Filter.h_Filter_sw.c</mark> _libImpulse_Makefil	

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 + 📃 🗖	×
Name Applications Places System 😪 🏵 👘	3))
meixu@localhost:/home/meixu/Xilinx/win_share_5	×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp	
BOOTAS arch/powerpc/boot/fixed-head.o	
BOOTCC arch/powerpc/boot/ep88xc.o	
BOOTCC arch/powerpc/boot/ep405.0	
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 -0 dtb -o arch/powerpc/boot/virtex440-ml507.dtb -b 0 -p 1024 /home/meixu/Xi	
linx/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"	
<pre>WRAP arch/powerpc/boot/simpleImage.virtex440-ml507</pre>	
<pre>WRAP arch/powerpc/boot/simpleImage.initrd.virtex440-ml507</pre>	1
rm arch/powerpc/boot/virtex440-ml507.dtb	E B
[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  $\rightarrow$  Lauch 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.

Watch the Linux boot display in the Tera Term window:

Fire Term - COMI VIT
File Edit Setup Control Window Help
initrd head: 0x1f8b0808
Linux/PowerPC load: console=ttyS0 root=/dev/ram rw ip=off
Finalizing device tree... flat tree at 0x7aa300
Using Xilinx Uirtex440 machine description
Linux version 2.6.28-rc6 (rootPlocalhost.localdomain) (gcc version 4.2.2) #37 PR
EEMPT Wed Jan 21 18:28:09 PST 2009
Found initrd at 0xc05a9000:0xc079c19e
Zone PFN ranges:
DMA 0x000000000 -> 0x00010000
Normal 0x000100000 -> 0x000100000
Movable zone start PFN for each node
early\_node map[1] active PFN ranges
0: 0x000000000 -> 0x000100000
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 0xfdfff000
PlD hash table entries: 1024 (order: 10, 40% bytes)
Clocksource: timebase multIa00001 shift[22] registered
Console: colour dummy device 80x25
Dentry cache hash table entries: 16384 (order: 4, 65536 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
met\_namespace: 636 bytes
NET: Registered protocol family 16
PC: Probing PCI hardware
NET: Registered protocol family 16
PC: Probing PCI hardware

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



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====================================	
> Done filtering two slots, execution time : 0.737921 seconds ====================================	
> 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

- Xilinx Open Source Linux Wiki: Configuring, Building and Loading Linux After 9/1/08 <u>http://xilinx.wikidot.com/configuring-building-and-loading-linuxafter-9-1-08</u>
- 2. PowerPC Processor Reference Guide <u>http://www.xilinx.com/support/documentation/user\_guides/ug0</u> <u>11.pdf</u>
- 3. PowerPC 405 Processor Block Reference Guide <u>http://www.xilinx.com/support/documentation/user\_guides/ug0</u> <u>18.pdf</u>
- 4. Embedded Processor Block in Virtex-5 FPGAs Reference Guide <u>http://www.xilinx.com/support/documentation/user\_guides/ug2</u> <u>00.pdf</u>
- Accelerated System Performance with the APU Controller and XtremeDSP Slices <u>http://www.xilinx.com/support/documentation/application\_note</u> <u>s/xapp717.pdf</u>
- GNU Compiler Collection (GCC): Assembler Instructions with C Expression Operands <u>http://gcc.gnu.org/onlinedocs/gcc-</u> <u>3.4.3/gcc/Extended-Asm.html</u>
- 7. AltiVec Instruction Cross-Reference <u>http://developer.apple.com/hardwaredrivers/ve/instruction\_cros</u> <u>sref.html</u>
- 8. Impulse Support Forum: APU and Linux <u>http://impulse-</u> support.com/forums/index.php?showtopic=143&st=0&p=671