Port Embedded Linux to XUP Virtex-II Pro Development Board

ZHOU Qingguo[1,2*], YAO Qi[1,2], LI Chanjuan[1,2], Hu Bin[2,3]
1. Distributed & Embedded System Lab (DSLab), Lanzhou University, China;
2. Engineering Research Center of Open Source Software and Real-Time Systems, Ministry of Education, Lanzhou University, China
3. Department of Computing, Birmingham City University, UK
yq000cn@gmail.com; *zhouqg@lzu.edu.cn; lichanjuan04@lzu.cn

Abstract

As its free, rich code resource and supporting many kinds of CPU architecture, Linux is making steady progress in the embedded arena. The Virtex-II Pro serials development system, produced by Xilinx company, provides an advanced hardware platform that consists of a high performance Virtex serials platform FPGA surrounded by a comprehensive collection of peripheral components that can be used to create a complex system and meet different special embedded application areas. This paper mainly introduces porting embedded Linux to the XUP Virtex-II Pro development system, and using serials of development tool kits. It is a good guide to develop embedded system with FPGA development system and Linux.

1. Background

Embedded system is a special computing system, application-centric, based on computer technology. With customizing the hardware and software, embedded system can meet different situations constrained by function, reliability, cost, size, power consumption. Generally, embedded system includes common embedded system based on DSP or MCU, Application-specific Integrate Circuits (ASIC) [1], and System on a Programmable Chip(SoPC). SoPC has flexibility of common processor, and ASIC's high speed and reliability, so it meets complex applications in a variety of occasions and is becoming a bright spot in the embedded application field. Xilinx Virtex-II Pro serials development system are integrated two PowerPC405 cores, which have high performance and low cost. Added with reconfigurable FPAG, the system is very flexible and powerful.

With the embedded processor performance enhancement and kinds of application requirements, it is urgent to need embedded operation system (EOS) to improve the performance of whole system. Embedded Linux is a good choice, because there are many advantages with Linux, such as configuration flexibility, great portability, open source, supporting most of architecture, etc. So it is a great try to combine Xilinx FPGA development board integrated two PowerPC405 cores, with embedded Linux operation system to enhance the performance of embedded system.

978-1-4244-3930-0/09/$25.00 ©2009 IEEE

2. Introduction

The whole system development depends on both designing the hardware and software. The main development flow is shown in figure 1.

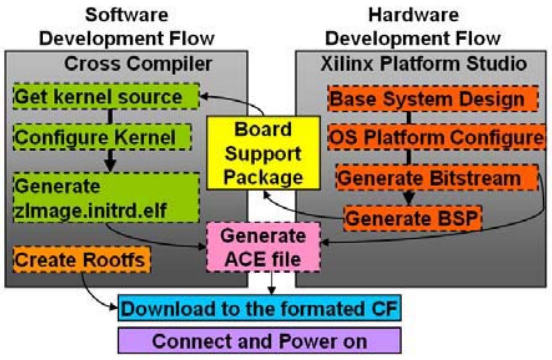


Figure 1. System Development Flow

2.1.Virtex-II Pro development system

The XUP Virtex™-II Pro FPGA development system can be used at virtually any level of the engineering curricula, from introductory courses through advanced research projects.

Key Features [2]:

- Virtex-II Pro XC2VP30 FPGA with 30,816 Logic Cells, 136 18-bit multipliers, 2,448 Kb of block RAM, and two PowerPC Processors
- DDR SDRAM DIMM that can accept up to 2Gbytes of RAM
- 10/100 Ethernet port

- USB2 port (for FPGA configuration only)
- Compact Flash for FPGA configuration and data storage
- PS/2 and RS232 port

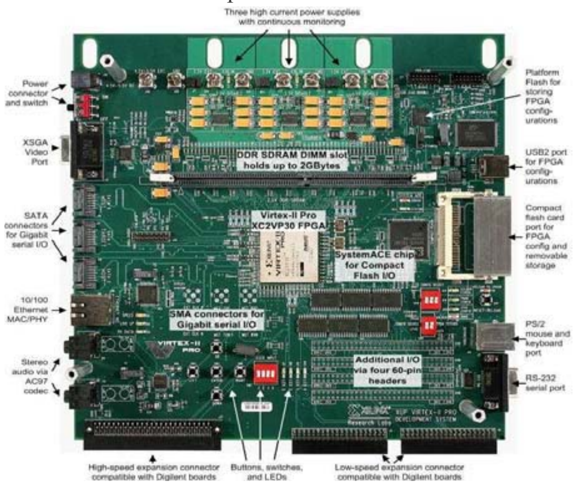

Three high current power supplies with continuous monitoring
Power connector and switch
XSGA Video Port
SATA connectors for Gigabit serial I/O
10/100 Ethernet MAC/PHY
Stereo audio via AC97 codec
Platform Flash for storing FPGA config-urations
USB2 port for FPGA config-urations
Compact flash card port for FPGA config and removable storage
PS/2 mouse and keyboard port
RS-232 serial port
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
High-speed expansion connector compatible with Digilent boards
Buttons, switches, and LEDs
Low-speed expansion connector compatible with Digilent boards

Figure 2. XUP Virtex-II Pro Development System

Xilinx provides Xilinx Platform Studio(XPS)[3], which is a suite of design tools, for developing the hardware and software components of an embedded processor system.

2.2 Embedded Linux system

Embedded Linux development broadly involves three tiers: the bootloader, the Linux kernel, and the root filesystem.

Bootloader is a piece of code that will be executed for initializing devices and booting kernel before starting operation system. Xilinx has developed a bootloop[4] program, located at the processor's boot location, to initialize the processor, seemly like bootloader.

The Linux kernel is an operation system kernel used by a family of Unix-like operating systems. And it needed to reduce the functions to fit the given development environment.

A root filesystem is one kind of filesystem which is used for the storage, organization, manipulation, and retrieval the data, and it should generally be small, since it contains very critical files.

3. Base system design

We use Xilinx Platform Studio 9.1i to build the base system with PowerPC 405.

3.1. Base system building

In this step, we build the base system with Base System Builder Wizard, supplied by XPS, to choose processor, peripherals, bus specification, Block RAM and so on. Full system should be customized to meet performance, functionality, and cost goals. The base system is shown in figure3.

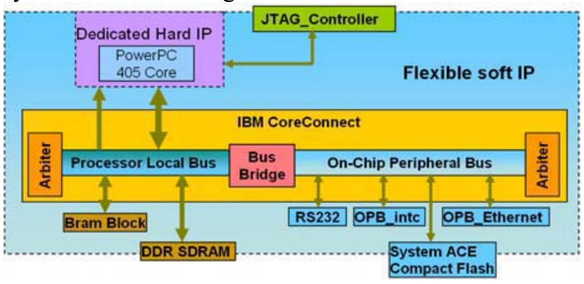


Figure 3. Base System Architecture

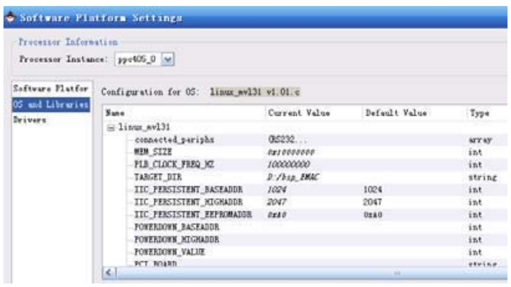
And the added IP cores are listed in table1.Aim at the actual application, some other IP cores, which are supported by Xilinx or created by yourself, could be added, such as VGA, PS/2, GPIO.

Table 1. The major IP cores' parameter

IP Core Name	Parameter
PPC405	300MHz
OPB_UART16550	Use interrupt
OPB_SystemACE	Use interrupt
OPB_Ethenet_MAC	No DMA and use interrupt
PLB_DDR_256MB	256MB and use interrupt
PLB_BRAM_IF_CNTLR	128KB

3.2. OS platform configure

The main purpose of this part is setting the OS platform environment to automatically generate board support package(BSP)[5], including booting code, Xilinx device drivers and OS initialization program. The Software Platform Settings table is shown in Figure 4.



Processor Instance: ppc405_0

Configuration for OS: linux_mvl31 v1.01.c

```
Name                          Current Value   Default Value   Type
⊟ linux_mvl31
  connected_periphs           (RS232...                       array
  MEM_SIZE                    0x10000000                      int
  PLB_CLOCK_FREQ_HZ           100000000                       int
  TARGET_DIR                  D:/bsp_EMAC                     string
  IIC_PERSISTENT_BASEADDR     1024            1024            int
  IIC_PERSISTENT_HIGHADDR     2047            2047            int
  IIC_PERSISTENT_EEPROMADDR   0xA0            0xA0            int
  POWERDOWN_BASEADDR                                          int
  POWERDOWN_HIGHADDR                                          int
  POWERDOWN_VALUE                                             int
  PCI_BOARD                                                   string
```

Figure 4. Software Platform Settings Table

3.3. Generate bitstream and BSP

First, marked the bootloop application for ppc405_0, function like bootloader, to initialize BRAMs[6]. Then choose buttons on toolbar the following orders: Generate bitstream, Build All User Applications, and Update bitstream with software program information. If some IP cores you add, are created by yourself, you should program corresponding drivers for embedded Linux system.

If all are managed to generate, bitstream, named download.bit, which is used to configure FPGA, is located in the Platform Studio project directory under implementation/, and BSP(see figure5) is located in TARGET_DIR directory, which is set in Software Platform Settings table.

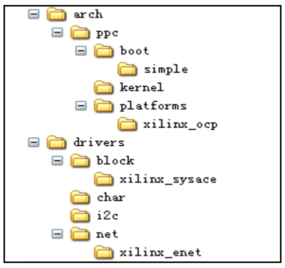


Figure 5. Generated BSP directory structure

4. Generate operation system

The host machine and target machine have different processor architecture, so it is necessary to build cross-compiling environment.

4.1. Build cross-compiling environment for PPC

Thanks to Dan Kegel[7], who builds crosstool[8] for easily building cross-compiling environment. Download crosstool tarball from [9] into Linux OS. Some install commands are listed below:

```
#tar –zxvf crosstool-0.43.tar.gz
#cd crosstool-0.43
```

Edit demo-powerpc-405.sh file and amend similar content like that:

```
TARBALLS_DIR=$HOME/crosstool-0.43
RESULT_TOP=/opt/crosstool
eval `cat powerpc-405.dat gcc-3.3.6-glibc-2.3.2.dat`
sh all.sh –notest

#sudo mkdir /opt/crosstool
#sudo chown $USER /opt/crosstool
#sh demo-powerpc-405.sh
```

When it finishes, cross-compiling toolchain is located under /opt/crosstool/gcc-3.3.6-glibc-2.3.2.

4.2. Getting the kernel sources

The full Linux source tree supporting Xilinx board is not popular. So you need to install bkf command on Linux box to get the ppc-kernel-2.4 development tree.

```
#wget http://www.bitmover.com/bk-client2.0.shar
#sh bk-client2.0.shar
#cd bk-client2.0
#sh demo.sh
#./bkf clone bk://ppc.bkbits.net/linuxppc_2_4_devel linuxppc-2.4.26
```

Copy the board supported package (BSP) into kernel source correspondingly. Different version of XPS will generate different BSP, so some source code in kernel should be mended. The patch[10] for kernel source dose a lot of work. It is very important, or some errors will happen when compiling the kernel.

4.3. Set system environment

Set the host machine Linux system environment path including the cross-compiling toolchain's directory.

```
#export PATH=/opt/crosstool/gcc-3.3.6-glibc-2.3.2/powerpc-405-linux-gnu/bin:$PATH
```

4.4. Configure and compile kernel

Change directory into kernel source.

```
#make menuconfig
```

The main module options are listed:

- Platform support
  Processor Type:40x
  Machine Type:Xilinx-ML300
  TTYS0 device and default console:UART0
- General setup
  Networking support: Enable
  Default bootloader kernel arguments: Enable
  Initial kernel command string: " console = ttyS0, 38400 ip=on root=/dev/xsysace/disc0/part3,rw "
- Block Devices
  Xilinx on-chip System ACE: Enable
  RAM disk support: Enable
  Default RAM disk size:16384
- Character devices
  Standard/generic(8250/16550 and compatible UARTs) serial support: Enable
- File systems
  Virtual memory file system support: Enable
  Second extended fs support: Enable

#make dep && make zImage.initrd

After that, the compiled kernel image named zImage.initrd.elf is generated at arch/ppc/boot/images directory.

4.5. Build root filesystem

The kernel needs a root filesystem to mount at startup, and a root filesystem must contain everything needed to support a full Linux system, such as:

- Basic directories: /dev,/proc,/bin,/etc,/lib,/usr
- Basic set of utilities: sh, ls, cp, mv, **etc**.
- Basic set of config files: rc, inittab, fstab, **etc**.
- Devices: console, ttyS0, **etc**.

Thanks to Klingauf[11] who writes a script mkrootfs[12], combined with busybox[13] to make building root filesystem easy. Download the mkroofs from[12] and busybox from [13], and decompress the files to your home directory.

#cd mkroofs

Modify some parameters (listed as following) in mkrootfs.sh:

- LFS= the prefix location of your rootfs
- CC= cross compiler
- TARGET_PREFIX=location of cross toolchain library file
- BUILD_TOOLS= directory of cross toolchain
- PPC_KERNEL= directory of kernel source
- PPC_KERNEL_VERSION=2.4.26
- Replace "busybox-1.00-pre2" with your busybox directory

#sh mkrootfs.sh

Then the generated rootfs is located at the directory which "LFS" is set.

5. Download and test

The System Advanced Configuration Environment (System ACE) [14] is a configuration solution using compact flash for systems with multiple FPGAs.

5.1. Build ACE file

The System ACE file can be used to configure the FPGA, initialize BRAM, initialize external memory with Linux kernel elf file, and boot up the processor in system. Generate System ACE file following the listed steps.

Edit genace.opt file, the content as following:

```
-jprog
-board user
-target ppc_hw
-hw download.bit
-elf zImage.initrd.elf
-configdevice devicenr 1 idcode 0x1127e093 irlength 14 partname xc2vp30
-debugdevice devicenr 1 cpunr 1
-ace system.ace
```

Put download.bit, zImage.initrd.elf and genace.opt together at a directory, the Launch EDK shell(one kind of XPS tool) and execute commands after change directory into the above three files' location.

#xmd –tcl genace.tcl –opt genace.opt

At the end, System ACE file named system.ace is created.

5.2. Format Compact Flash

The Compact Flash (CF) is used to store System ACE file and root filesystem. And System ACE only supports FAT12 and FAT16 file system, so the CF need formatted like that (see table2).

Table 2. Compact Flash partition list

Partition	Type	Size
1	FAT16	32MB
2	Linux Swap	256MB
3	Linux(ext3)	remain

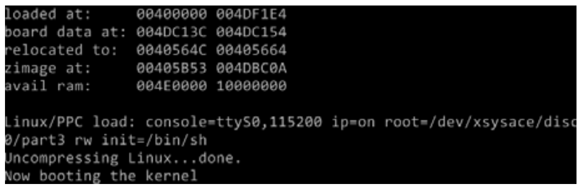
5.3. Download and test

Copy the System ACE file into partition 1 of CF, and copy all files of root filesystem into partition 3 of CF. Then put the CF and 265MB SDRAM into corresponding slots of development board, connect board and host machine with serial port cable, and connect LAN with network cable.

Configure serial communication software (such as Putty in Windows):

- Set the given COM Port
- Set Baud rate: 38400, corresponding with kernel configuration.

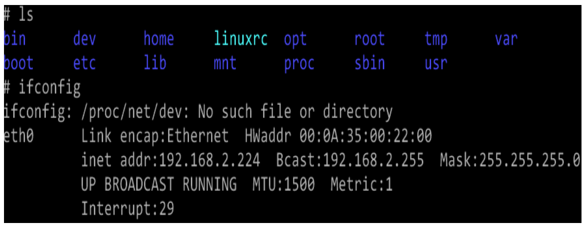
Boot the system. See figure 6 and figure 7.



```
loaded at:     00400000 004DF1E4
board data at: 004DC13C 004DC154
relocated to:  0040564C 00405664
zimage at:     00405B53 004DBC0A
avail ram:     004E0000 10000000

Linux/PPC load: console=ttyS0,115200 ip=on root=/dev/xsysace/disc
0/part3 rw init=/bin/sh
Uncompressing Linux...done.
Now booting the kernel
```

Figure 6. System booting information



```
# ls
bin      dev      home     linuxrc  opt      root     tmp      var
boot     etc      lib      mnt      proc     sbin     usr
# ifconfig
ifconfig: /proc/net/dev: No such file or directory
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:00
          inet addr:192.168.2.224  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          Interrupt:29
```

Figure 7. Entering into Linux system

It means I managed to port Linux kernel to the FPGA development board.

6. Conclusion and future

The whole development environment is sort of complicated, including Linux and Windows operation system, and kinds of development tools. So it costs much more time to master the whole development flow. Even so, kinds of great tool kits offer powerful designing functions and make the whole system much more flexible.

This paper guides you to familiar with the whole development flow of embedded system with FPGA and embedded Linux before carrying out the actual application.

In the future, according to the given application environment, design peripheral IP cores and interface, and add them into FPGA, then develop the corresponding drivers in Linux. Hardware and software co-design will become popular with the continuous development of the theory and technology.

7. Acknowledgements

Thanks to DSLab, Lanzhou University, for supplying the XUP Virtex-II Pro development board. Also thanks all members in DSLab, who give me many suggestions.

8. References

[1] http://en.wikipedia.org/wiki/ASIC

[2] http://www.xilinx.com/products/devkits/XUPV2P.htm

[3] http://www.xilinx.com/ise/embedded/edk_pstudio.htm

[4]http://www.xilinx.com/itp/xilinx9/help/platform_studio/html/ps_p_dld_initializing_bitstreams_bootloops.htm

[5] Rick Moleres, Milan Saini. "Generating Efficient Board Support Packages", *Embedded Magazine*, 2006.3:23-27.

[6]http://www.xilinx.com/support/documentation/ipembedprocess_memoryinterface_block-ram.htm

[7] http://www.kegel.com/crosstool/

[8] http://www.kegel.com/crosstool/current/doc/crosstool-howto.html

[9] http://kegel.com/crosstool/crosstool-0.43.tar.gz

[10]http://www.cs.washington.edu/research/lis/empart/files/xup_patch_linuxppc_tree.tgz

[11] http://www.klingauf.de/v2p/index.shtml

[12] http://www.klingauf.de/v2p/mkrootfs.tgz

[13] http://www.busybox.net/

[14]http://www.xilinx.com/support/documentation/data_sheets/ds080.pdf