ePICS

Engineering Proprioception
in Computing Systems

Project no. 257906
EPiCS
Engineering Proprioception in Computing Systems

A Guide to building the ReconOS/Linux reference system

Project acronym:	EPiCS	
Project name:	Engineering Proprioception in Computing Systems	
Call and Contract:	FP7-ICT-2009-5	
Grant agreement no.:	257906	
Project duration:	2010-09-01 – 2014-08-31 (48 months)	
Coordinator:	UPB	University of Paderborn (DE)
Partners:	IMPERIAL	Imperial College London (UK)
	UIO	University of Oslo (NO)
	UNI-KLU	University of Klagenfurt (AT)
	UOBIRM	University of Birmingham (UK)
	EADS	EADS Innovation Works (DE)
	ETHZ	ETH Zürich (CH)
	AIT	Austrian Institute of Technology (AT)

This project is supported by funding from the FET proactive initiative "Self-Awareness in Autonomic Systems" by the European Union 7th Framework Programme.

Document History

Revision	Date	Scope of changes	Description	Implemented by
0.1	2011-09-01	All sections	New document	UPB

Contents

1 A guide to building the ReconOS/Linux reference system

This section will show you how to build a virtual memory enabled ReconOS/Linux system. Since the automated ReconOS tool flow is geared towards building under the eCos operating system, large parts of the hardware design have to be assembled manually. Because this process is prone to errors, we recommend starting with our ReconOS/Linux reference design and change it according to your application needs. The following sections will explain how this can be done.

1.1 Prerequisites

This tutorial will assume that you already have checked out the ReconOS repository and build all required tools and libraries. This tutorial assumes that you have a Xilinx XUPV2P board with a compatible memory module installed.

1.2 Building the reference design

In order to build the reference design we first build the hardware which results in a bitstream for the FPGA. Then, we build the necessary software which includes the Linux kernel, the OSIF module, the ReconOS user space library and the application binary.

Building the hardware

First, change to the project directory:

```
$ cd $(RECONOS)/demos/shared_tlb_demo
```

A makefile in the project directory provides a convenient way to generate the static bitstream. Using the makefile the bitfile and the device tree file can be generated using the following two lines:

```
$ make setup
$ make hw
```

Building the software

Next, we compile the Linux kernel using the device tree file generated in the previous step. This is automated using the makefile in the project directory:

```
$ make linux
```

Unlike the ReconOS package in eCos, the ReconOS extensions to the Linux system software are split into two parts: A kernel module (`osif.ko`) and a user space library (`libreconos.a`). The library can be build using the Makefile in the project directory:

```
$ make libreconos
```

The next step is to configure and compile the Linux kernel. Since our system was developed against kernel version 2.6.27-rc9 we recommend to use that version. The kernel sources can be obtained from the Xilinx Linux repository located at `git://git.xilinx.com`. Once you have obtained the kernel sources and applied the ReconOS/Linux kernel patch, create a link in your project directory pointing to the kernel sources:

```
$ ln -s linux <path to patched kernel tree>
```

Alternatively you can also copy the kernel tree into your project directory. After configuring the kernel you can use the makefile to build the kernel image:

```
$ make linux
```

After the kernel is configured and an image has been build, you can now create the ReconOS kernel module, located at `$(RECONOS)/core/linux/drivers/reconos`. The final step in the build process is the generation of the software executable:

```
$ make shared_tlb_demo
```

This command creates the software demo application and links it to the `libreconos` library.

1.3 Running the test application

After the hardware design and the software components have been build, we have to prepare our host computer system to export the reference design root file system and to connect to the Linux console via the serial link.

Setting up the NFS root file system

In order to set up the NFS root file system export, we have to create a virtual network interface on the host computer. The IP address of the new interface has to be the same address that is specified in the reference design kernel. In our case this is the address `192.168.30.1`. To set up the virtual network interface, execute the following command (may require root privileges).

```
$ ifconfig eth0:1 192.168.30.1 up
```

Next, we have to set up NFS. Open the file `/etc/exports` and add the following line:

```
/exports 192.168.30.0/255.255.255.0(rw,insecure,no_root_squash,\
no_all_squash,no_subtree_check)
```

Now create the directory `/exports/rootfs` and copy the reference design root file system into that new directory. Connect the RS232 and Ethernet ports of the FPGA board with the host computer and program the FPGA with the bitstream created above. Start `minicom` and connect to the PowerPC with `xmd` and download the kernel image:

```
$ xmd
XMD% connect powerpc hw
XMD% dow linux/arch/powerpc/boot/simpleImage.virtex405-xup-opb-eth-tft-cf.elf
XMD run
```

You should now see the boot messages of the kernel on the minicom console. After the kernel boot process finished and the root file system is mounted you are greeted with a shell prompt and should be able to list and browse the root file system.

Loading the OSIF drivers

Before you can use the demo application you have to load the ReconOS kernel module. Copy the `osif.ko` kernel module that was created in a previous step to the NFS root file system.

```
$ cp osif.ko /exports/rootfs
```

Then switch to the minicom console and load the module:

```
$ insmod ./osif.ko
```

Then you can run the application:

```
$ ./shared_tlb_demo 1024
```

1.4 Making changes to the project

While building and setting up the system the first time can be quite time consuming, making changes to existing applications or creating new ones is easy and straight forward. After making changes to the software part of the application, it is sufficient to recompile it and put the new version into the NFS root file system. When you change the hardware threads, you only have to create the new bitstream and program the FPGA. A recompilation of the Linux kernel is only necessary when you make changes to the hardware design (i.e., adding or removing peripherals or changing bus addresses).