

Xiang Li

March 1, 2009

*This page is intentionally left blank*

# Contents

<b>B A Step-by-Step Instruction to Repeat the Thesis Project</b>	<b>5</b>
B.1 Hardware / Software Develop Environment . . . . .	5
B.1.1 DE2-70 Board . . . . .	6
B.1.2 A RS232 to USB Cable . . . . .	6
B.1.3 An Ethernet Cable . . . . .	7
B.1.4 A Speaker or an Earphone . . . . .	7
B.1.5 A PC . . . . .	7
B.1.6 Cygwin . . . . .	7
B.1.7 OpenRISC Tool Chain . . . . .	8
B.1.8 Quartus II . . . . .	9
B.1.9 The Thesis Zip File . . . . .	10
B.2 Step-by-Step Instructions . . . . .	10
B.2.1 Quartus Project and FPGA Programming . . . . .	10
B.2.2 Download Software Project by Bootloader . . . . .	13
B.2.3 Download Music Data and Play . . . . .	16
References . . . . .	21

*This page is intentionally left blank*

## Appendix B

# A Step-by-Step Instruction to Repeat the Thesis Project

This is a step by step instruction shows how to repeat our thesis project. For those people who are interested, it should be easier to reproduce the same simple MP3 player as we did by following this instruction.

### B.1 Hardware / Software Develop Environment

Below is a list of the hardware devices and software tools used for the thesis project. If you can manage to get the same develop environment, it would be very helpful to repeat the project.

- DE2-70 Board
- A RS232-to-USB Cable
- An Ethernet Cable
- A Speaker or an Earphone
- A PC
- Cygwin
- OpenRISC Tool Chain
- Quartus II
- Our Thesis Zip File

### B.1.1 DE2–70 Board

Perhaps the most important hardware needed for the thesis project is a DE2–70 FPGA board. The DE2–70 is a Development and Education board based on ALTERA’s Cyclone II FPGA (EP2C70). It is produced by Terasic. From their website you can find all the information to the DE2–70 [1]. The board costs 599 USD, or 329 USD for academic users, which is not cheap but luckily many universities have already had lectures or labs based on this board. So if you are a student, maybe you can try to ask your professor or laboratory to borrow one. Just like we did before started our thesis.

Someone may have Terasic’s DE2 board instead. It is possible to port the thesis project from the DE2–70 to the DE2, because the 2 boards are very similar. The main difference between the DE2 and the DE2–70 is that the DE2 uses Cyclone II EP2C35 as the FPGA, which has less on-chip resources than EP2C70. Besides, the DE2 has only 512KB SSRAM and 8MB SDRAM on board, while the DE2–70 has 2MB SSRAM and 64MB SDRAM. However these resources on the DE2 are already enough for our thesis project. I am sure that our project can be ported to the DE2 after some necessary modifications, although I didn’t ever try that.

Some other people may have different FPGA boards, like Terasic’s DE1 or maybe even a Xilinx FPGA board. In these cases, I will have to say “I wish I could help, but . . .” The reason that our project cannot fit into this FPGA boards is mainly because those boards probably do not have the audio CODEC WM8731 to play music or the DM9000A to support an Ethernet connection. This makes the project porting becomes too difficult or even impossible. But it is still a good topic to try some other projects related to the OpenRISC processor on all kinds of FPGA boards.

### B.1.2 A RS232 to USB Cable

A RS232 / USB cable sets up a UART connection between a PC and the DE2–70 board. So we can communicate to the OpenRISC processor by any of your preferred serial terminal software<sup>1</sup>. Most of PCs do not have RS232 ports nowadays. That’s why it needs to convert from RS232 to USB. But if you do have a PC with the RS232 port, a normal RS232 cable works fine too. The RS232 to USB cable is easy to find. For example if you go to amazon.com and search “RS232” + “USB”, many results return, and any of them could work.

---

<sup>1</sup>By the way, if you don’t have a preferred terminal software at the moment, we enclosed one in the thesis project zip file, under the folder `/tools/uart_terminal/`.

Another important reason to have a UART connection is that we designed a bootloader which uses the RS232 port to download program data from the PC to the DE2-70's SSRAM / SDRAM. Because we don't have a programmer or debugger for OpenRISC to load the binary file, and the ALTERA's USB Blaster cable can be only used to program the FPGA, perhaps the easiest way to get the program data into on-board external RAM is by the bootloader<sup>1</sup>.

### B.1.3 An Ethernet Cable

An Ethernet cable, or called Category 5 cable [2], connects the DE2-70 to the PC's network adapter, i.e. network card. So we can create a UDP connection to transfer music data to the board. Usually this cable is easy to get, unless your PC uses wireless network only.

### B.1.4 A Speaker or an Earphone

To hear the music, at least an earphone is needed. If you want to play it aloud, try to get a speaker instead.

### B.1.5 A PC

Of course, we need a PC. Our PC for the project is running 32-bit Windows XP SP3 as operating system. I guess Windows VISTA, whether 64-bit or not, should work too.

### B.1.6 Cygwin

Cygwin is a Linux-like environment for Windows [3]. I like this software. It is very good to provide a Linux-like environment, and it is very small if compare to other virtual machine software (e.g. VMare). The reason we need a Linux environment is because the OpenRISC processor tool chain is derived from GNU tool chain, i.e. GCC, GNU Binutils, GDB etc. To compile source codes into binary files for the OpenRISC processor we have to use these tools, which are not so friendly to Windows.

---

<sup>1</sup>Actually there is another tricky way which is by Terasic's "control panel", but that is not as handy as our bootloader.

Then why not just use a native Linux PC? Hmmm, this is a good question. Actually we did try to use CentOS as the operating system at the beginning, but I gave up soon. The official excuse is that we need ALTERA's Quartus to design the FPGA project, and we had some unenjoyable experience with the Quartus Linux edition. But to be honest, the real reason is that I don't know Linux well and usually got stuck by some very basic operations, like how to install software under Linux. So finally I switched to Windows / Cygwin, which made me more productive. For Linux pros, the OS should not be a problem, so of course you may try out our project under a Linux PC, after recompiling everything that we have compiled in Cygwin.

If you want to use Cygwin, there is something you need to know. Cygwin is good, but the installation of this software is however kind of complicated, because it asks you to choose the components to be installed from a quite long list. And each time when you reinstall the Cygwin, or want to copy an exact same Cygwin environment to another PC, this process becomes boring. If sometimes—just like the experience I had—you forget a package to download and install, and happen to get a place without Internet, it could be a good time to feel the word “crazy”.

To fix this trouble, I spent some time looking for help, and now there is a solution. The file “installed.db” under the folder `/cygwin/etc/setup/` is actually a list of the name of the packages have been installed. If you somehow get an installed.db and make it stored in the same path, when running the “setup.exe” the system will display those packages as installed. Then you can simply choose to reinstall all these packages, which will create a Cygwin environment exactly as the installed.db file specified. See this webpage for more information [4]. My installed.db is uploaded to my Blog too [5], which you can take as a reference when your Cygwin is not working properly.

### **B.1.7 OpenRISC Tool Chain**

The OpenRISC Tool Chain converts high level programming language, like C, into binary instruction codes which OpenRISC processor understands.

When I was a beginner, I feel the most difficult part of the thesis project, is how to get a working tool chain, i.e. if you write a C program, how to make it to become binary instructions, how to download it to the board, how to simulate and debug the codes. Because most of the commercial CPUs nowadays, for example ALTERA's NIOS, have already provided an IDE which including everything. Just by several clicks, all the compiling, downloading and debugging staffs are done. But in the OpenRISC world without IDEs,

you will have to experience all these difficulties in person.

As said before, the OpenRISC tool chain is modified based on the GNU tool chain, which is totally free software under GPL. The OpenRISC tool chain developers well performed their duties as the GPL asked. They provides the tool chain source codes on the CVS of the opencores.org which can be downloaded freely. There are also instructions which guide you to set up these tools on your PC. If you are a Linux pro and hardworking, this shouldn't be tough for you.

Unfortunately I am not a Linux pro and I am lazy, so I tried but failed to compile a working tool chain under Cygwin. The tool chain compiled by myself is always not working properly or efficiently. Luckily, the OpenRISC teams provided a pre-compiled tool chain package for Cygwin. Just by unzipping the package to the Cygwin system path, you will get all the tools for OpenRISC software developing. This tool chain package is still available from the Opencors.org website [6]. We also included a copy in the thesis zip file.

However this pre-build tool chain package for Cygwin is quite out of date, although it is still working. From the file name it says the package was made at 2003, i.e. 6 years ago. Almost at the same period when I did the thesis, the OpenRISC team made a great update to the OpenRISC tool chain. I haven't examined what they did exactly yet, but of course there should be lots of changes. So I sincerely hope that my readers, if anyone happens to read to this sentence, go to try out the new tool chain on OpenRISC's webpage [7], instead of using the pre-build Cygwin package.

### B.1.8 Quartus II

The Quartus II is the FPGA development software designed by ALTERA. Because the DE2-70 board uses ALTERA's FPGA, the Quartus becomes the one cannot be replaced.

I used Quartus II 8.0sp1 Web Edition in the thesis, with the web license that is freely to apply from [www.altera.com](http://www.altera.com) [8]. So no money needs to be paid for the FPGA development tools. Now most of EDA vendors provide an "evaluation" version of their tools, but if I remembered correctly it was the ALTERA who started this, that also the reason why I keep using Quartus for years.

### B.1.9 The Thesis Zip File

Oh, don't forget to get our project zip file if you want to try our thesis project. This file can be downloaded at my Blog [5], which includes both hardware and software projects, as well as some tools and other stuff. I promise this file will be downloadable all the year 2009, but no guarantee after that.

## B.2 Step-by-Step Instructions

Now let's get started the step by step instructions of how to repeat the thesis project. Basically there are 3 big steps:

1. Review the Quartus project and program the FPGA on the DE2-70.
2. Download the software project to the DE2-70 by the bootloader.
3. Run the software to receive music data from PC and play on the board.

### B.2.1 Quartus Project and FPGA Programming

- 1.1 Start Quartus II and open the Quartus project in the `/hardware` folder.
- 1.2 The top level entity is in `/hardware/component/top/orpXL_top.vhd`. The entity includes all the pins which are allocated on the EP2C70. Figure 1 shows the file.
- 1.3 As we can see, all the modules used in the project are saved in different folders under `/hardware`. There is one module needed to be mentioned a little.

The `/ram` folder contains an on-chip RAM module. It is configured as 64KB. The `/ram/ram0.mif` in the same folder is the data file will be written into the RAM when the FPGA is programmed. In our thesis zip file, this `ram0.mif` contains the data of the bootloader. So if you do not change this file, the bootloader will be downloaded into the board at the time when programming the FPGA.

It is possible to replace this `ram0.mif` with something else. For example, you may write your own program, covert it to MIF format with our `ihex2mif` tool in the folder `/tools/ihex2mif`. By this way you can run any of your programs on the OpenRISC platform as long as

```

components/top/orpXL_top.vhd
16  library ieee;
17  use ieee.std_logic_1164.all;
18
19  entity orpXL_top is
20  port (
21      clk_50:    in    std_logic;
22      rst_n:    in    std_logic;
23
24      -- SSRAM interface
25      sram_a:    out   std_logic_vector (18 downto 0);
26      sram_dq:   inout std_logic_vector (31 downto 0);
27      sram_adsc_n: out  std_logic;
28      sram_adsp_n: out  std_logic;
29      sram_adv_n: out  std_logic;
30      sram_be_n: out   std_logic_vector (3 downto 0);
31      sram_ce1_n: out  std_logic;
32      sram_ce2:  out  std_logic;
33      sram_ce3_n: out  std_logic;
34      sram_clk:  out  std_logic;
35      sram_dpa: inout std_logic_vector (3 downto 0);
36      sram_gw_n: out  std_logic;
37      sram_oe_n: out  std_logic;
38      sram_we_n: out  std_logic;
39
40      -- SDRAM interface
41      dram0_a:  out   std_logic_vector (12 downto 0);
42      dram0_d:  inout std_logic_vector (15 downto 0);

```

Figure B.1: Top level entity of the project

the program is smaller than 64KB. However if the program is getting bigger than the limit, the bootloader is needed anyway which can load the program into the external 2MB SSRAM. That is large enough for most embedded programs.

If the default ram0.mif is modified and you want the bootloader back, rename the file:

/tools/program\_loader/server\_openrisc/proloader\_server.mif  
to ram0.mif and copy it into the /ram folder.

P.S. If you only want to update the MIF file without making other hardware modifications, the whole FPGA project recompilation in Quartus is not necessary. Just choose to update the MIF and run the assembler to generate a new SOF file again: short-key in Quartus II is Alt+R+U then Alt+R+A+A.

- 1.4 Now it is time to equip the DE2-70 and get all the cables connected: the power cable, the USB cable for FPGA programming, the USB to Serial cable and the Ethernet cable.

Please note that, don't connect the speaker to the board for now. Because the DE2-70's built-in demonstration project plays a high frequency sine wave when power-on, the noise will hurt your ear if the speaker is connected. Unless you turn the switch off before put on the power. The Switch 17 of the board is used to control whether the sine wave played or not. To switch off the noise you need to turn the switch

up. When saying “up” it means you need to push the switch closer to the LEDR17.

- 1.5 Now the board is ready, please program the FPGA with orpXL\_top.sof file in Quartus, like Figure 2 showed.

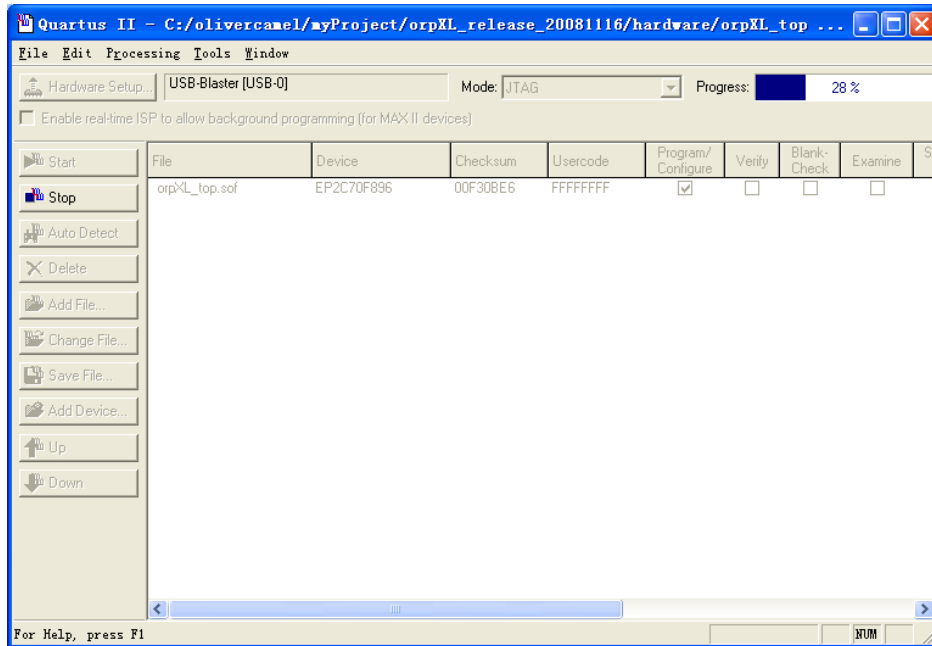


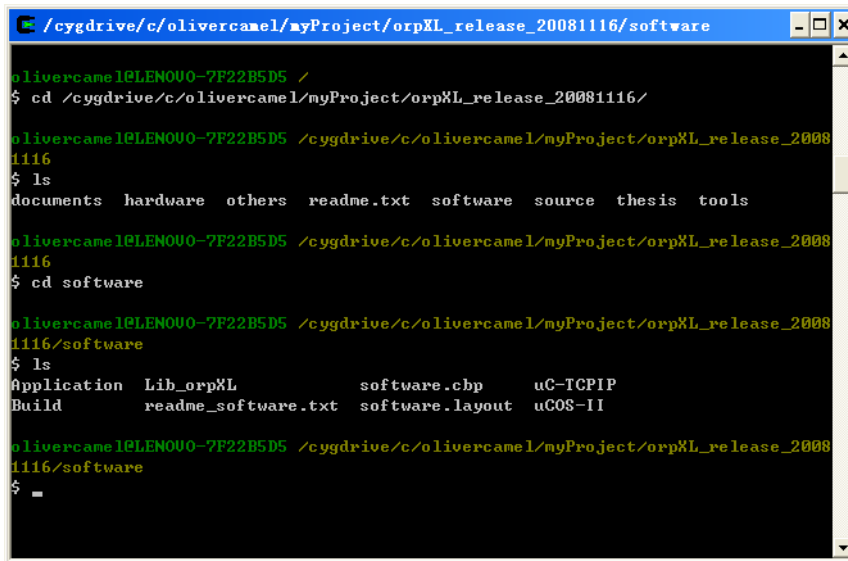
Figure B.2: FPGA programming

- 1.6 By programming the FPGA, the OpenRISC hardware platform will be downloaded to the board, and the bootloader will be placed into the on-chip RAM. Meanwhile the default DE2-70 demonstration project will be overwritten, so the since ware noise is gone. From now on don't be afraid to connect the speaker anymore.
- 1.7 In the project, we used the Switch 17 as the reset key of the hardware system. When the Switch 17 is pulled down, it means the reset is on. And when the Switch 17 is pushed up, the system starts working.

After the FPGA is programmed, please reset the hardware system by putting the Switch 17 down, and then up. After the switch is pushed up, the bootloader will start running. It stays in an endless loop waiting while reading the RS232 port, so the data could be transferred from PC via serial connection.

## B.2.2 Download Software Project by Bootloader

- 2.1 Start Cygwin, and enter the folder of the software project, i.e. /software. Figure 3 shows the folder structure.



```
olivercaml@LEN000-7F22B5D5 /
$ cd /cygdrive/c/olivercaml/myProject/orpXL_release_20081116/

olivercaml@LEN000-7F22B5D5 /cygdrive/c/olivercaml/myProject/orpXL_release_20081116
$ ls
documents  hardware  others  readme.txt  software  source  thesis  tools

olivercaml@LEN000-7F22B5D5 /cygdrive/c/olivercaml/myProject/orpXL_release_20081116
$ cd software

olivercaml@LEN000-7F22B5D5 /cygdrive/c/olivercaml/myProject/orpXL_release_20081116/software
$ ls
Application  Lib_orpXL          software.cbp      uC-TCPIP
Build        readme_software.txt software.layout   uCOS-II
```

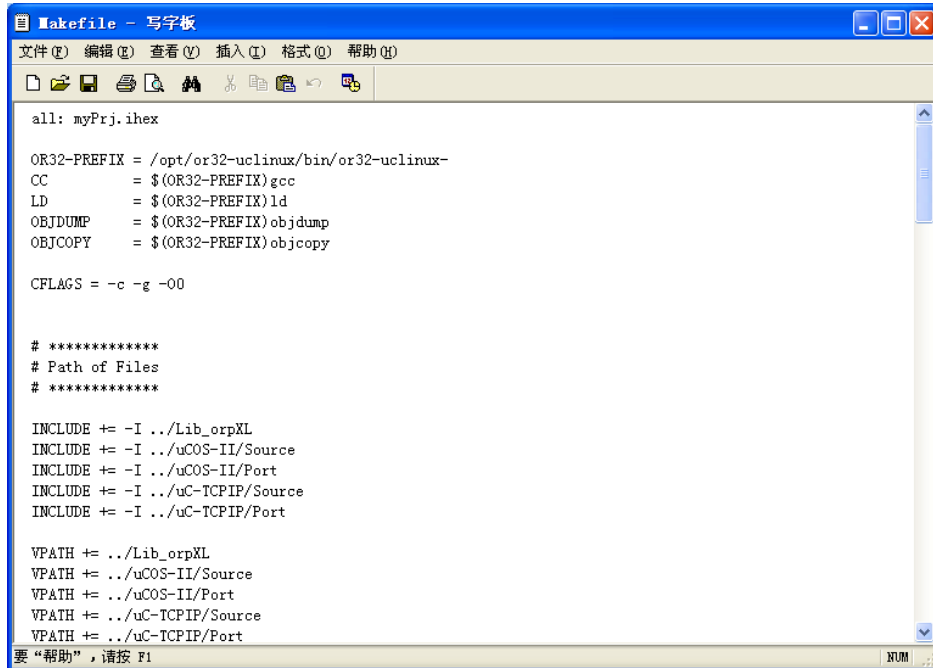
Figure B.3: Software project

- 2.2 The /software/build is the folder used to compile the software project.

To save time without input all the commands every time, a makefile script is made for the “Make” tool. Please examine the makefile that if the OpenRISC tool chain is placed in the right path. Otherwise those commands will not work. The makefile script is showed in Figure 4.

- 2.3 Now let’s recompile the software project. This is not necessary but interesting to try.

In the /build folder, run command “make all clean”, as showed in Figure 5. You will get a myPrj.ihex at the end of compilation. It is an Intel HEX format file which will be downloaded to the board by the bootloader. Also you will get a myPrj.dis. This is a disassembly file which shows all the instructions of the project. It is very helpful to check the disassembly file and understand what your software is actually doing.



```

all: myPrj.ihex

OR32-PREFIX = /opt/or32-uclinux/bin/or32-uclinux-
CC          = $(OR32-PREFIX)gcc
LD          = $(OR32-PREFIX)ld
OBJDUMP     = $(OR32-PREFIX)objdump
OBJCOPY     = $(OR32-PREFIX)objcopy

CFLAGS = -c -g -O0

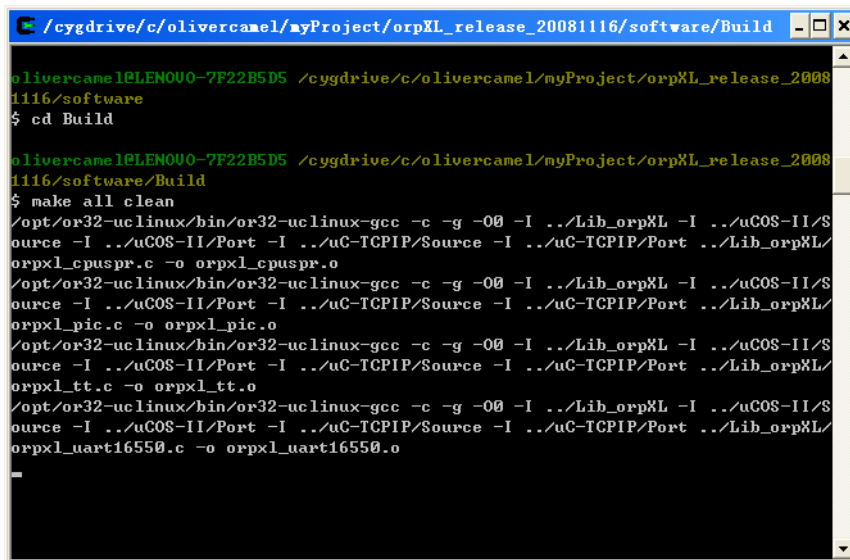
# *****
# Path of Files
# *****

INCLUDE += -I ../Lib_orpXL
INCLUDE += -I ../uCOS-II/Source
INCLUDE += -I ../uCOS-II/Port
INCLUDE += -I ../uC-TCPIP/Source
INCLUDE += -I ../uC-TCPIP/Port

VPATH += ../Lib_orpXL
VPATH += ../uCOS-II/Source
VPATH += ../uCOS-II/Port
VPATH += ../uC-TCPIP/Source
VPATH += ../uC-TCPIP/Port

```

Figure B.4: Makefile script



```

/cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
olivercanel@LENOVO0-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software
$ cd Build

olivercanel@LENOVO0-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
$ make all clean
/opt/or32-uclinux/bin/or32-uclinux-gcc -c -g -O0 -I ../Lib_orpXL -I ../uCOS-II/Source -I ../uCOS-II/Port -I ../uC-TCPIP/Source -I ../uC-TCPIP/Port ../Lib_orpXL/orpXL_cpuspr.c -o orpXL_cpuspr.o
/opt/or32-uclinux/bin/or32-uclinux-gcc -c -g -O0 -I ../Lib_orpXL -I ../uCOS-II/Source -I ../uCOS-II/Port -I ../uC-TCPIP/Source -I ../uC-TCPIP/Port ../Lib_orpXL/orpXL_pic.c -o orpXL_pic.o
/opt/or32-uclinux/bin/or32-uclinux-gcc -c -g -O0 -I ../Lib_orpXL -I ../uCOS-II/Source -I ../uCOS-II/Port -I ../uC-TCPIP/Source -I ../uC-TCPIP/Port ../Lib_orpXL/orpXL_tt.c -o orpXL_tt.o
/opt/or32-uclinux/bin/or32-uclinux-gcc -c -g -O0 -I ../Lib_orpXL -I ../uCOS-II/Source -I ../uCOS-II/Port -I ../uC-TCPIP/Source -I ../uC-TCPIP/Port ../Lib_orpXL/orpXL_uart16550.c -o orpXL_uart16550.o

```

Figure B.5: Build software project

2.4 Now let's start the bootloader to download the software project into DE2-70 board.

The bootloader is comprised with 2 parts: a server that is already

running on the FPGA with OpenRISC processor, and a client will be started now who sends the data file in the PC.

The executable `/build/proloader_client.exe` is the bootloader client that we are talking about. It was compiled by Cygwin-GCC and thus can only run in the Cygwin. Run the following command under `/build` folder:

```
/proloader_client.exe -d /dev/com5 -f myPrj.ihex -p
```

The parameter “-d” specifies the RS232 ports on your PC, you can check which port is allocated in you PC by the Windows Device Manager as showed in Figure 6. In Windows you always write the RS232 port in the format of “`/dev/comX`” where X is the port number read from the Device Manager. But in Linux, the format will be usually like “`/dev/ttySX`”.

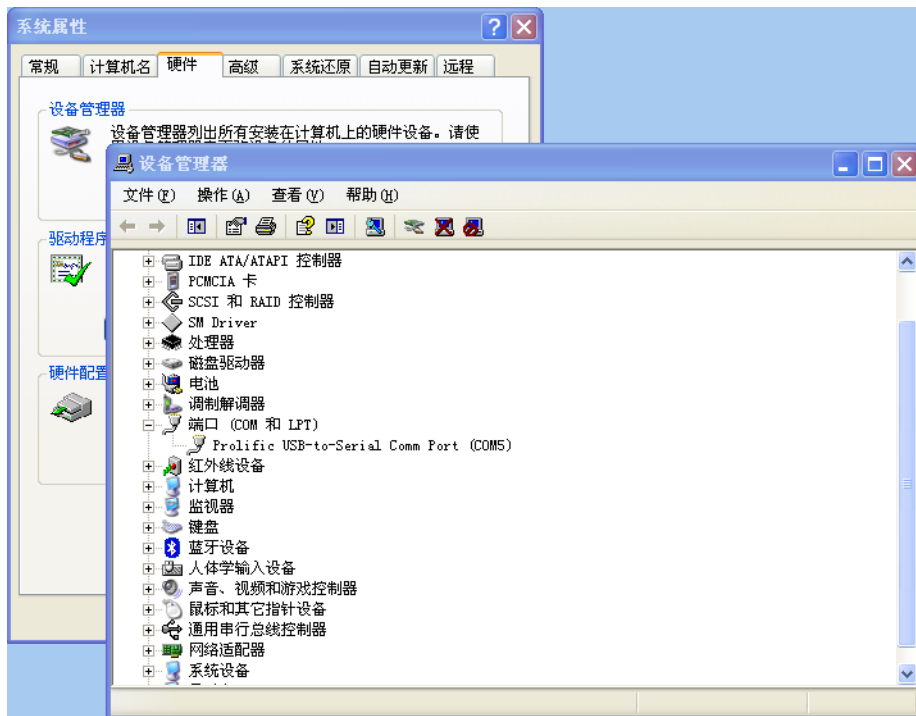


Figure B.6: Check USB-to-serial port ID

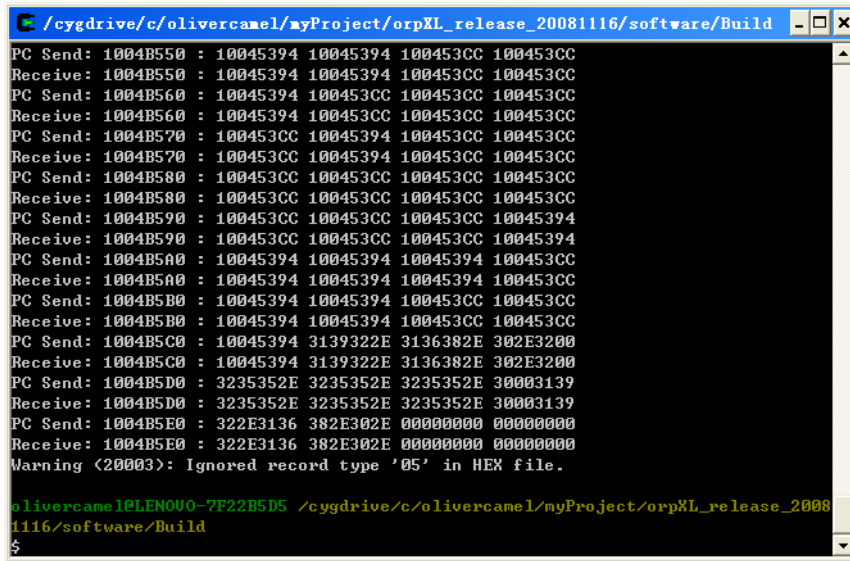
The parameter “-f” specifies the path of the HEX file.

The “-p” tells the bootloader to display the data being transferred in the Cygwin window. The reason to do this is because the loading time is always quite long<sup>1</sup>. To finish downloading the software project takes

<sup>1</sup>The speed of the serial connection is low and the bootloader is not designed very efficiently.

about 5 minutes. So the “-p” makes you be sure about that the system is still running. But the bad thing of the “-p” is that a lot of data will overwhelm and you screen will be flushed.

After the downloading of the thesis project HEX file, it will look like the Figure 7.



```

/cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
PC Send: 1004B550 : 10045394 10045394 100453CC 100453CC
Receive: 1004B550 : 10045394 10045394 100453CC 100453CC
PC Send: 1004B560 : 10045394 100453CC 100453CC 100453CC
Receive: 1004B560 : 10045394 100453CC 100453CC 100453CC
PC Send: 1004B570 : 100453CC 10045394 100453CC 100453CC
Receive: 1004B570 : 100453CC 10045394 100453CC 100453CC
PC Send: 1004B580 : 100453CC 100453CC 100453CC 100453CC
Receive: 1004B580 : 100453CC 100453CC 100453CC 100453CC
PC Send: 1004B590 : 100453CC 100453CC 100453CC 10045394
Receive: 1004B590 : 100453CC 100453CC 100453CC 10045394
PC Send: 1004B5A0 : 10045394 10045394 10045394 100453CC
Receive: 1004B5A0 : 10045394 10045394 10045394 100453CC
PC Send: 1004B5B0 : 10045394 10045394 100453CC 100453CC
Receive: 1004B5B0 : 10045394 10045394 100453CC 100453CC
PC Send: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
Receive: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
PC Send: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
Receive: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
PC Send: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Receive: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Warning <20003>: Ignored record type '05' in HEX file.

olivercanel@LEN000-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_2008
1116/software/Build
$

```

Figure B.7: Downloading and flushing finished

Usually the bootloader works fine without any problem, but I’ve had bad experience that occasionally the bootloader may stick at somewhere. And then the Windows XP gave a blue screen. I am not sure about the reason of this problem. Probably it is because some driver crashes. I have no idea how to avoid it. In case a blue screen comes out perhaps restarting the PC is the only thing we could do.

### B.2.3 Download Music Data and Play

3.1 Now the software project has been downloaded into the SSRAM of the DE2–70. Before starting the program, there are some configurations needed to do on your PC.

The first thing is to check your IP address. Please configure your IP address to 192.168.0.3, IP mask to 255.255.255.0 and the default gate to the 192.168.0.1, as showed in Figure 8. By the way, the IP address of the DE2–70 board is set to 192.168.0.2, where you are going to send your UDP packets to. These addresses are chosen for no reasons. You can of course change them to some other addresses in the source code.

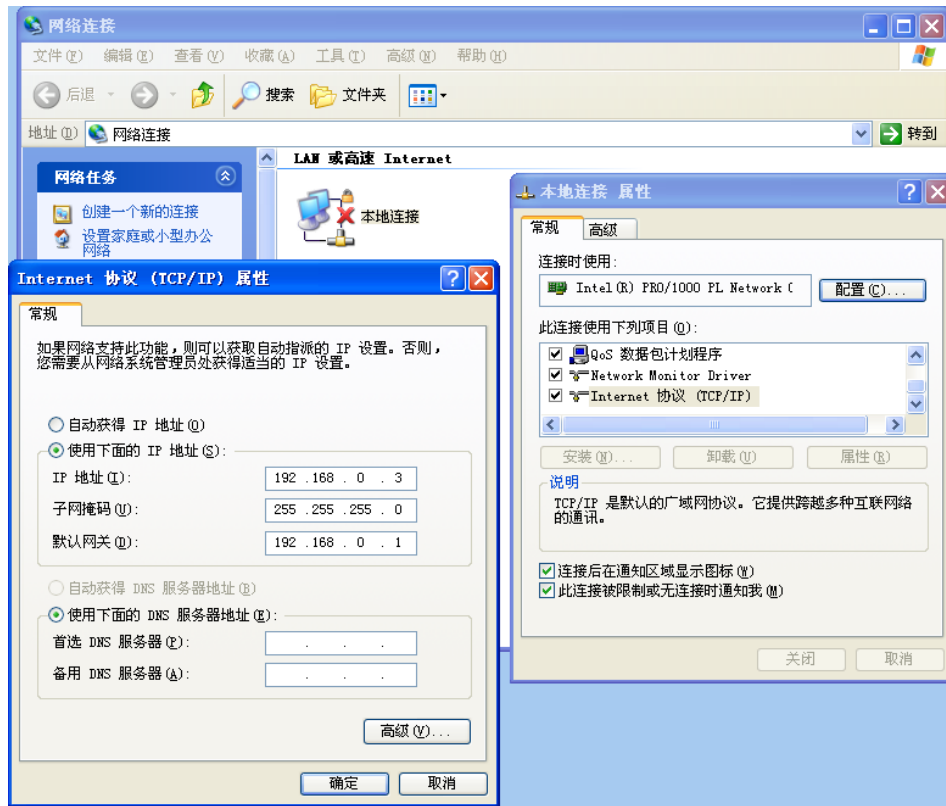


Figure B.8: IP Configuration

3.2 After that please disable all other network adapters on your PC if you have more than one. For example, my laptop has 2 network cards. The one is a wireless network card, and the other is a normal 100/1000Mbps network adapter. In this case, please disable the wireless network card and only keep the one which is connecting to the DE2-70.

3.3 Meanwhile please close all the software that may send TCP/IP packets to the Internet, like IE, MSN, and the anti-virus software that may auto upgrade themselves at certain time.

The steps 3.1-3.3 make sure that there will be only one program (the music player of our thesis) sending UDP packets to the only target address (your DE2-70 board). The reason is because our thesis application is not so reliable to handle all kinds of packets. If somehow another software broadcasts a UDP packet during the time we are downloading the music file and the board receives this packet, it will ruin all the data communications because our software does not know how to deal with it.

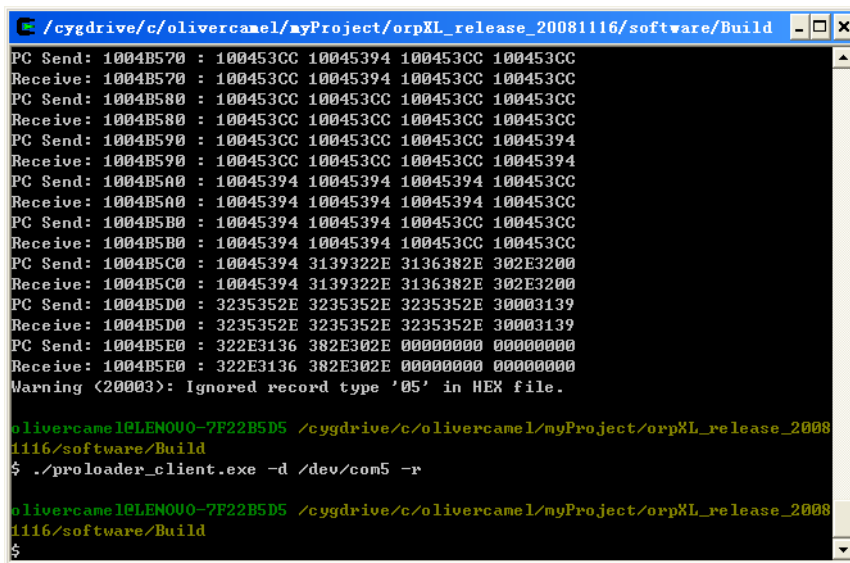
3.4 After the steps before have been checked. Start the music player that

we just downloaded through the bootloader again. It is a similar command but with other parameters:

```
./proloader_client.exe -d /dev/com5 -r
```

Still the “-d” specifies the RS232 port. The “-r” here tells the bootloader to jump to the entry point of the external SSRAM. So the program stored at there starts working and the job of the bootloader is done.

If the program starts without any problem, you will see the LAN is connected, showed in Figure 9 and 10.



```

/cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
PC Send: 1004B570 : 100453CC 10045394 100453CC 100453CC
Receive: 1004B570 : 100453CC 10045394 100453CC 100453CC
PC Send: 1004B580 : 100453CC 100453CC 100453CC 100453CC
Receive: 1004B580 : 100453CC 100453CC 100453CC 100453CC
PC Send: 1004B590 : 100453CC 100453CC 100453CC 10045394
Receive: 1004B590 : 100453CC 100453CC 100453CC 10045394
PC Send: 1004B5A0 : 10045394 10045394 10045394 100453CC
Receive: 1004B5A0 : 10045394 10045394 10045394 100453CC
PC Send: 1004B5B0 : 10045394 10045394 100453CC 100453CC
Receive: 1004B5B0 : 10045394 10045394 100453CC 100453CC
PC Send: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
Receive: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
PC Send: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
Receive: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
PC Send: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Receive: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Warning (20003): Ignored record type '05' in HEX file.

olivercanel@LENOVO0-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
$ ./proloader_client.exe -d /dev/com5 -r

olivercanel@LENOVO0-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
$

```

Figure B.9: Start software project via bootloader



Figure B.10: Ethernet connected

3.5 Now I want to spend some texts to explain the reset switch and the bootloader.

In our project, the Switch 17 is used to reset the system. When a reset is needed, push the Switch 17 firstly down, and then up.

Because the default starting address is pointed to the on-chip RAM, every time it is the bootloader that starts after a reset. To actually reset the software application, just run the command we did in step

3.4 again to let the bootloader jump to the external RAM. If the power of the DE2–70 board is not turned off, the data stored in the FPGA and the external SSRAM will always be valid. So there is no need to re-program the FPGA and download the software project on every resets.

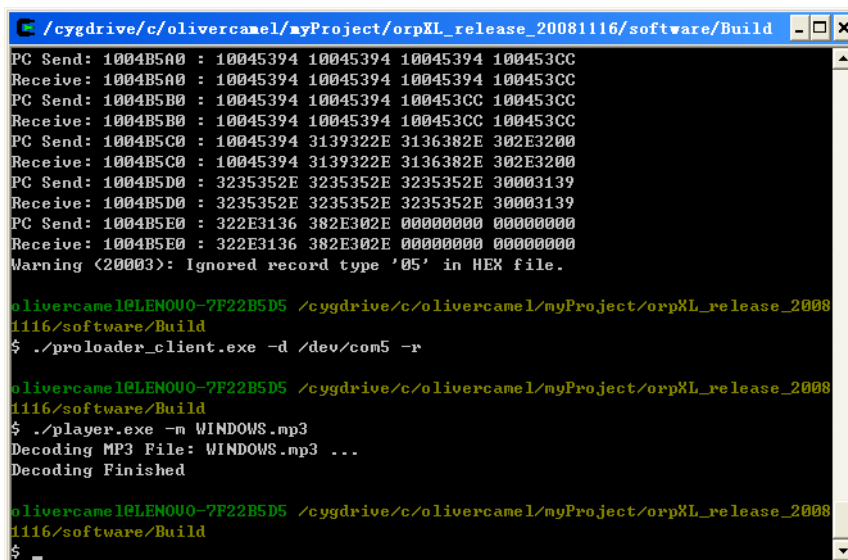
- 3.6 We used 4 7-segment LEDs in our project. Each 7-segments plus the digit can display 8 bits. So they are organized to show the value of 2 16-bit counters. On the DE2–70 board, the HEX0 and HEX1 is the first counter. Its value means how many valid UDP packets have been received. The HEX2 and HEX3 is the other counter which shows the number of invalid UDP packets received.

The software application is designed to check UDP packets all the time. So after the music player is started, you should see the HEX2 and HEX3 counter is changing its value.

- 3.7 The `/software/build/WINDOWS.mp3` is MP3 file used as the demo in our project. It is a very small MP3 file which only lasts 7 seconds.

First convert the MP3 file to WAV format. This is done by another application—`player.exe`, which is a client working on the PC who decodes MP3 file into WAV format by LibMAD and then sends the music data to DE2–70. For LibMAD, check their website for more information [9].

The command is: `./player.exe -m WINDOWS.mp3`. “-m” parameter specifies the path of the MP3 file. This is showed in Figure 11.



```

/cygdrive/c/olivercanel/myProject/orpXL_release_20081116/software/Build
PC Send: 1004B5A0 : 10045394 10045394 10045394 100453CC
Receive: 1004B5A0 : 10045394 10045394 10045394 100453CC
PC Send: 1004B5B0 : 10045394 10045394 100453CC 100453CC
Receive: 1004B5B0 : 10045394 10045394 100453CC 100453CC
PC Send: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
Receive: 1004B5C0 : 10045394 3139322E 3136382E 302E3200
PC Send: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
Receive: 1004B5D0 : 3235352E 3235352E 3235352E 30003139
PC Send: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Receive: 1004B5E0 : 322E3136 382E302E 00000000 00000000
Warning (20003): Ignored record type '05' in HEX file.

olivercanel@LEN000-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_2008
1116/software/Build
$ ./proloader_client.exe -d /dev/com5 -r

olivercanel@LEN000-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_2008
1116/software/Build
$ ./player.exe -m WINDOWS.mp3
Decoding MP3 File: WINDOWS.mp3 ...
Decoding Finished

olivercanel@LEN000-7F22B5D5 /cygdrive/c/olivercanel/myProject/orpXL_release_2008
1116/software/Build
$

```

Figure B.11: Decode MP3 file

- 3.8 After that, there will be a “temp.wav” generated in the `/build` folder too. It is 1.28MB while the MP3 is only 123KB. This is the file that going to be split up into UDP packets and transferred to the board.

Why not sending the MP3 file directly? No we didn’t make it, because in that case we need the LibMAD working with the OpenRISC processor on the DE2-70 too. This is not so difficult to implement because the LibMAD is written in ANSI C, but it uses several C standard Lib functions like `malloc()`, which we have no time to make it work on our hardware platform.

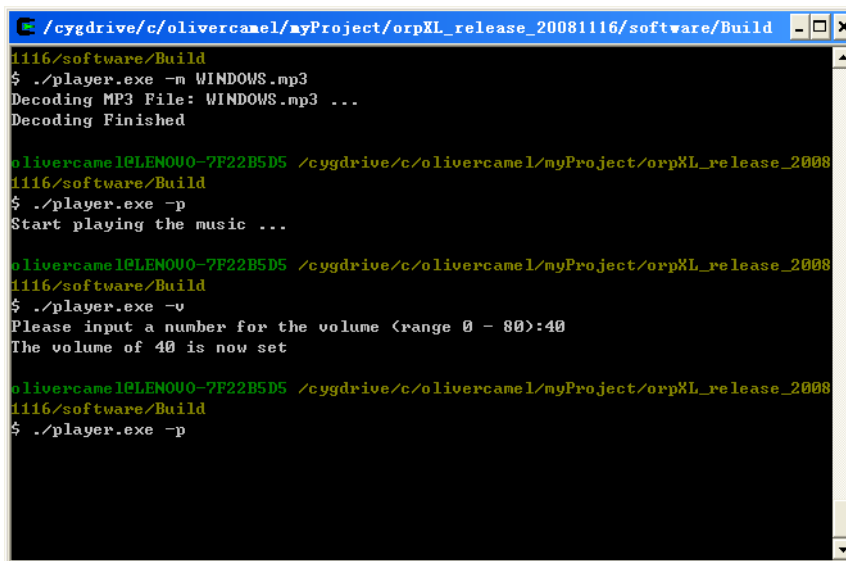
- 3.9 Download the temp.wav into the DE2-70 board. This will be placed into the 64MB SDRAM.

The command is: `./player.exe -w temp.wav -d`. “-w” specifies the path of the WAV file. “-d” tells the system to download. After this command is performed, you should see the counter on the DE2-70, i.e. the 7-segment HEX1 and HEX0 start counting.

In theory, you can download any other WAV files smaller than 64MB, but you probably won’t because the downloading speed is too slow. We can achieve only 3KB stable UDP speed on our platform. This may be enough to open a webpage but not capable for music files. You can actually calculate the time needed yourself. We are downloading a 1.28MB WAV files, and with only 3Kbytes per second. The time is more than stand up and walk around in the room for a while.

The low speed is because of multiple reasons: first the CPU is running at only 50MHz, and more importantly the platform, both the hardware and software, is not designed so efficient. There are lots of optimizations we could do, but there just have no more time.

- 3.10 When the WAV file is downloaded successfully, you should see like Figure 12. But the downloading process could go wrong if it is interfered by other software on your PC. In case the HEX0 and HEX1 stop changing its value but the “Downloading finished” doesn’t show up, you will need to restart the system and try it again. If unfortunately this step always fails, you may need some software tool like Wireshark [10] to monitor your TCP/IP packets and check what’s wrong exactly.
- 3.11 Before play the music, please lower down the volume with the command: `./player -v`. The software will ask to input a value between 0-80. A value between 30 to 60 would be fine and at here we just fill in 40. The default value is set to 80, which is a mistake. The too loud sound will hurt your ear if you are wearing an earphone. Also sometimes the WM8731 works not properly when it is set to 80.



```
cygdrive/c/olivercamel/myProject/orpXL_release_20081116/software/Build - _ x
1116/software/Build
$ ./player.exe -m WINDOWS.mp3
Decoding MP3 File: WINDOWS.mp3 ...
Decoding Finished

olivercamel@LEN000-7F22B5D5 /cygdrive/c/olivercamel/myProject/orpXL_release_2008
1116/software/Build
$ ./player.exe -p
Start playing the music ...

olivercamel@LEN000-7F22B5D5 /cygdrive/c/olivercamel/myProject/orpXL_release_2008
1116/software/Build
$ ./player.exe -v
Please input a number for the volume (range 0 - 80):40
The volume of 40 is now set

olivercamel@LEN000-7F22B5D5 /cygdrive/c/olivercamel/myProject/orpXL_release_2008
1116/software/Build
$ ./player.exe -p
```

Figure B.12: Set volume and play the music

3.12 Finally type the “play” command: `./player.exe -p`. If everything is fine, you should hear the music by now. Cheers!

## Reference:

- [1] Website, Terasic Technologies, <http://www.terasic.com.tw/>, Last visit: 2009.03.01.
- [2] Webpage, *Category 5 cable*, from Wikipedia, [http://en.wikipedia.org/wiki/Category\\_5\\_cable](http://en.wikipedia.org/wiki/Category_5_cable), Last visit: 2009.03.01.
- [3] Website, Cygwin, <http://www.cygwin.com/>, Last visit: 2009.03.01.
- [4] Webpage, A discussion on how to duplicate Cygwin environment, <http://cygwin.com/ml/cygwin/2008-04/msg00100.html>, Last visit: 2009.03.01,  
Don't forget to read all the reference posts and follow-ups.
- [5] Webpage, The page of my thesis publication, [http://www.olivercamel.com/post/master\\_thesis.html](http://www.olivercamel.com/post/master_thesis.html),  
Last visit: 2009.03.01,  
This is where you can find my installed.db of the Cygwin.
- [6] Webpage, OpenRISC 1000: OLD GNU Toolchain Port, [http://www.opencores.org/projects.cgi/web/or1k/gnu\\_toolchain\\_port\\_old](http://www.opencores.org/projects.cgi/web/or1k/gnu_toolchain_port_old),

Last visit: 2009.03.01,

You can download the old toolchain package compiled for Cygwin at here.

- [7] Webpage, OpenRISC 1000: NEW GNU Toolchain Port, [http://www.opencores.org/projects.cgi/web/or1k/gnu\\_toolchain\\_port](http://www.opencores.org/projects.cgi/web/or1k/gnu_toolchain_port), Last visit: 2009.03.01.
- [8] Website, ALTERA, <http://www.altera.com/>, Last visit: 2009.03.01.
- [9] Website, underbit technologies, <http://www.underbit.com/products/mad/>, Last visit: 2009.03.01, This is where you can find the information of the LibMAD.
- [10] Website, WireShark, <http://www.wireshark.org/>, Last visit: 2009.03.01, Wireshark is the world's foremost network protocol analyzer.