ECE532 Digital Hardware

Individual Report

Prepared by: Christopher Yip 990917941

Table of Content

1. Introduction

This project uses the Xilinx Multimedia board and the MicroBlaze embedded processor to function as a music player similar to Apple's iPod mp3 player. This project was built based on last year's Guitar Project and AC97 controller project to include playback control and storage of recorded audio. The audio format used is 16-bit PCM data.

The main storage medium is the Compact Flash (CF) slot, accessed through the Xilinx System ACE interface. Music can therefore be recorded to and/or retrieved from CF media. Additionally, it also features storage and playback using on-board ZBT RAM in case if compact flash is not present.

The playback control status of the system will be displayed using the RS232 serial CONPORT connection connected to a computer screen. It will tell the user if the system is playing back data, recording data and control audio processing options.

The main goal of this project is to show that it is possible to use the microprocessor to access the Compact Flash hardware to store and retrieve data. Access to the Compact Flash resource increases the number of uses for this board. For example, it can act as a network storage device by incorporating TCP/IP functionalities in the microblaze soft processor.

2. What was done

The list of tasks that were done will be explained in this section. We divided the project into tasks that are responsible by either Ben and I. Some tasks are responsible by both of us since we only have one board to work with.

2.1 Work breakdown

The following are the descriptions of a list of tasks Ben and I were responsible for:

Ben:
- Propose scope of the project and break down tasks
- Task scheduling, and ensuring things complete on time
- Administrative book-keeping of tasks done.
- Worked on hardware Filter core in Matlab and simulations in Modelsim
- Sourced information on Compact Flash
- Set up hardware base system for the project and a test system for the Compact Flash

Chris:
- Write software code for CF and debugging
- Figure out how to format the compact flash
- Write software code for sound processing

- Write software code for User Interface
- Simulation of Standalone FSL in Modelsim
- Create a simple user defined core to test FSL on Hardware
- Implement and simulate an echo core

Ben and Chris:
- Define scope of the project and separate tasks
- System integration of software code
- Updated AC97 project from 6.2 to 6.3 and debugged project to get it working
- Integrated simple user defined core to test FSL on Hardware
- Integrated and debug FIR filter core
- Integrated and tested echo core

2.2 Hurdles Encounter during the Project

This is the first time I have the chance to develop an embedded system. My previous experiences were mostly related to developing individual core. Because of this, a lot of time was spent understanding how different blocks are hooked up and how to use the software. Most of the problems that I encountered in the beginning of the project were simply related to hardware configuration problems, such as not hooking up the reset signal probably, not connecting to the right bus, etc. However, towards the later phase of the project, problems encountered were much harder to solved. I will describe some problems that we encountered in this section.

The most frustrating aspect related to the EDK tool was the GDB debugger. I used it frequently since I was responsible for most of the software components in the project. After running a single debug session, the debugger would refuse to execute the code. To get around this problem, I had to reprogram the board after every debug session.

One of the problems I encountered was storing and reading data from a compact flash. The first problem was getting the compact flash card to format properly. I followed the instruction posted on the Xilinx website http://support.xilinx.com/xlnx/xil_ans_display.jsp?iLanguageID=1&iCountryID=1&getPagePath=14456 to format the compact flash in order for the Sysace controller to recognize the card. However, I was unable to get the error status led of the Sysace controller to turn off no matter how I formatted the card. Fortunately, I was able to format the CF properly using a Canon digital camera.

After solving the problem with formatting a compact flash, I proceeded with the development of the software that reads and writes to the compact flash. I started off using the XilFatfs library, but I only managed to create an empty file on the card. The driver crashed when I tried to write data to the file. I reason of using XilFatfs first was that it creates a FAT on the CF which allows read/write to CF easily. I then tried to use the xsysace drivers, but I did not get that to work either. At the end, I went back to using the XilFatfs library and figured out what's causing the driver to crash. Instead of calling

the higher level API, sysace_fwrite() and sysace_fread(), I used read_sector() and write_sector() instead.

2.3 Things I learnt about the tools

2.3.1 XPS

I learnt about the backend of the tool and tricks to make the tool do to what you want. For example, changing the VHDL files in pcore does not force the tool to re-synthesize the core. The tool only re-synthesizes the core only if the system.mhs file is modified. It would actually be nice to have an option to specify which core(s) to be re-synthesized instead of re-synthesizing the whole system whenever the system.mhs is changed.

The GDB debugger does not work very well in the version of XPS that I used. If I have already debugged once, I need to reprogram the FPGA before I can start the debugger again.

2.3.2 ModelSim

When I tried to simulate the FSL, I only compiled one file – fsl_v20.vhd and ModelSim did not complain. However, the simulation did not work. I learnt that ModelSim might not complain even if some files are missing.

2.4 Design Flow

At the beginning of the project, we broke down our system into components. In addition, we designed our system with testability in mind. For example, we decided to connect the audio process core to the microblaze instead of connecting it directly to the audio core. The reason behind that is so that we can observe the data coming out of the audio core in software in case there are errors.

We simulated and tested each component before integrated them into a system. It made our debugging process a lot easier. We simulated the fir filter and the echo core with ModelSim, and we wrote simple software to test them on hardware.

We tried to implement and test a basic system before moving to a more complex system. We tried to have a backup plan for most of the modules we developed in order to mitigate the risk. In our case, we put together an audio recording system using ZBT memory first, in case we could not manage to get the compact flash to work. Also, we implemented a backup echo core which we used since our FIR filter core did not work on hardware.

Since we have a small group, we did not use any version control software during our project.

2.5 Modules Developed

I developed the following modules:

- Audio echo core in VHDL
- system.c – contains C code for UI, sound processing, saving/loading from CF, interacting with AC97 core and echo core

The behavior of the echo core was simulated with ModelSim. The core is then tested on hardware using software which plays back a pre-recorded voice. The different functionalities in system.c were tested separately on the Microblaze:

- UI – observe the behavior of the software when a input is given
- Software sound processing – using the ac97 to record voice, process the sound with software, then playback. The output sound was observed.
- Save/load from CF – raw data were written to and read back from the CF. The buffer was cleared before any read operation to ensure all available data was read from the CF.

2.6 What I Learned

This is the first time for me working with an embedded system and I learnt a lot during the course of the project. This is also the first time I build a system by putting different IP cores together and writing software to interact with those blocks. I also learnt how to use different commercial tools such as ModelSim, XPS, ISE, etc.

From this project, I learnt that being a good designer is not just writing code. But also be able to debug the system and understand the tool you use. I found that most of the time spent was actually debugging the system. The process was frustrating sometimes but I believe it was a valuable experience.

2.7 Community Contribution

We managed to write and read data from the compact flash. We have documented the steps to set up the sysace controller in our final report. We believe this will be beneficial to the group in the future who wish to use compact flash in their projects.

3. Feedback to Xilinx

I found the Xilinx tools very user friendly and the menus are organized in a logical manner. However, there is one suggestion that I have that I believe would improve the tool. Whenever I modified the .mhs file, all cores would get synthesized automatically, even the ones that were not changed. It would speed up the tool a lot if it only re-synthesized the cores that are changed.

In terms of documentations, I was having problems finding the information I needed on the Xilinx website. Also, it would be useful if Xilinx can provide more examples for the multimedia board, e.g. code that uses the compact flash.

4. Course Feedback

I think the project timeline worked well. We were given a few weeks to learn the software and what can be done with the board. But I wish that our group were allowed to sign out the board at the beginning of our project. A lot of the time, we were constrained by the working hours of the design centre. If we had our own board, we could have proceeded much faster.

I enjoyed the freedom I have in this course. The "open" lab concept was a great idea. Since most of the people in the class are working with an embedded system for the first time, I think it would be useful if we can have one-on-one with the TA every week to get feedback on our projects. It can make sure we are taking the right approach.

In terms of course material, I liked learning about the different FPGA technologies used by Altera and Xilinx. I would like to see more design examples related to computer hardware, for example, how PCI and DMA work.

The grading structure seems to concentrate too much on the report. It would be nice if the demo is worth more since it is the main part of the project. Also, I think the group report should worth more than the individual report since it contains more information about our project.