Individual Report

Prepared By: Benjamin Lai (991016994)

Table Of Contents

1. Introduction

The topic of my project is to build the Xpod. It is an imitation of an Apple iPod built using the Xilinx Multimedia board. Technical details of the entire system can be found in the group report.

My partner Chris Yip and I initially had no idea what project to do except that we both wanted to work with audio. We then came across one of the very first lectures in the course where we broke down the iPod into a system block diagram. That was the inspiration for the project.

We also thought that it would be interesting to see if the Compact Flash reader can be used to store data. As far as we know, no group in previous years has utilized the Compact Flash in their project. This way we would be able to contribute something to the ECE532 knowledge base. Perhaps someone next year may be able to further improve on what we started with.

2. What was Done

In this section I will describe the breakdown of tasks, what my duties were and how I accomplished them. This section also includes information about the methodology used to accomplish the tasks and issues that myself and the group came across.

2.1 Breakdown of tasks

Although Chris and I are taking very different courses, we still both have very similar course schedules. This means that it was very likely that when he had a break I also have a break. As a result we did a lot of the work together, helping each other out along the way.

Generally we decided to break down the task of the project into sub-tasks that we were each responsible for, to give things a bit of structure. Since Chris is very good with software coding it was decided that he would be responsible for the software aspect. Because I had more experience with hardware coding and VHDL/Verilog, I was in charge with the hardware aspect. I also took the role as the project administrator, to keep track of our progress and to make sure we don't fall behind. It was very easy to fall behind schedule since there are always other things that we need to do for our classes.

Below is a listing of what each person was "responsible" for. However this was not cast in stone, since from time to time we would have emergencies come up and we would help each other by doing some of the work:

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 Compacct 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

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

2.2 Hurdles Encountered During the Project

This was the first time that I have been exposed to the concept of what an embedded system is. The concept of embedding an entire microprocessor in an FPGA was so novel to me. But it makes so much sense since it has the advantage of fast prototyping and design time, not to mention the lower design costs.

Knowing nothing about what a MicroBlaze processor was or what XPS was, it took me some time to get a clear understanding of how all the hardware components interacted with the software and how the microprocessor is used. The lectures helped a lot but for me, I simply needed the time to assimilate the vast amount of information that was presented to me.

Another hurdle that I encountered was the issue of finding the right information on the website. For example, Professor Chow provided us with an application note, which discusses how to connect user-defined cores to the FSL bus. I was hoping that a similar document would be available to show me how to read and write to the Compact Flash. But instead there was nothing available. I ended up having to go through numerous documents (they are all listed in the Group Report) and find bits and pieces of the needed information. It was quite time consuming because this information was not centralized into one place.

Debugging with the software also posed a challenge. Sometime the error message that XPS reports are very cryptic. The "Answer Records" on http://support.xilinx.com either do not apply to the project or simply doesn't make sense. For example, when Chris and I were integrating the FIR filter core into the system, we kept getting an XST error complaining that we have an undefined filter core. We were certain that we had everything the software required. Yet the error never went away. I remembered from previous experience in ISE that, the netlist file of a CoreGen-generated core must also reside in the same directory as the XST project. For the system this was in the "Implementation" directory. After copying the filter netlist to this directory everything was fine.

At the administrative level, trying to keep track of the progress was a bit difficult and we seemed to always fall behind schedule. I think that the quality of the work schedule written at the beginning could make or break a project. For example if it is not detailed enough, then a certain task may actually take more time than expected. I was fortunate to have a hardworking partner who is on the top of his game. We were generally very good on keeping on track and things would get completed within the targeted amount of time.

2.3 Other Software Tools Used

A major part of my task in the project was to implement a hardware core to filter digital audio. A FIR or Finite Impulse Response filter is required. I set off to do a “bass boost” core to start, which ended up being the only core that I made because of time constraints. As a result other pieces of software were needed to get the job done.

I first defined what the bass boost function should do and decided that it should have about a 5dB gain in the low frequency range of 0Hz to 5KHz and 0dB gain for frequencies above 5KHz. MatLab was used to derive the coefficients of the FIR filter to achieve the bass boost function. Details on the core and the coefficients are found in the Group Report. Thanks to MatLab I was able to derive the coefficients quickly and accurately.

CoreGen, which is the IP Core generator that comes with the ISE environment, was used to generate a FIR filter instead of writing my own. The software interface takes the set of coefficients that were derived in MatLab in a specially formatted file (all documented in the data sheet for the core) and generates the appropriate logic. One issue that I was not able to solve was the fact that CoreGen seemed to accept only integer coefficient values. The MatLab derived coefficients are actually floating point numbers. As a compromise, I rounded the coefficients to the nearest integer value instead of keeping the decimal places.

ISE was used to generate the VHDL test bench for the FIR filter core as well as the associated tcl script to run it in ModelSim. This is done by first writing a wrapper design which instantiates a copy of the core and importing it into an ISE project. The directions on how to generate the appropriate test bench can be found in Tutorial Module 07 (version 6.3) located here: http://www.eecg.toronto.edu/~pc/courses/edk/modules/6.3/m07.pdf

2.4 What I learned About the Tools

2.4.1 ISE

When generating testbench waveforms in ISE (which will in turn generate a VHD or Verilog testbenches), it only allows you to simulate up to about 50000 clock cycles. Secondly, the Solaris version of ISE does not have the graphical waveform generation tool. In a nutshell, I used ISE to generate the basic testbench for the design. Then I manually edited the testbench file to create the desired waveforms and at the desired time.

ISE also provides an excellent HDL syntax checker tool. It is the last item XST flow. All that is needed to start syntax check is to double click on that icon.

2.4.2 ModelSim

ModelSim runs much quicker if it was installed on a local drive. In the Microprocessor labs on the 3rd floor of BA, ECF labs on the 1st floor of BA, Design Centre and as well as in the Solaris computers, ModelSim seems to be executed from a networked drive. This significantly slows down the compilation of files during the simulation.

2.4.3 XPS

While we were integrating the FIR filter into our system within XPS, I noticed that the software doesn't re-synthesize the VHDL core files even after editing them. In essence, it will only re-run everything if the MHS is updated. While trying to integrate the FIR Filter, we were constantly making updates to the wrapper VHDL file, which needed to be re-synthesized by XST. However there was no option available (or at least I couldn't fine one) that allows me to update only that one core. We had deleted all the hardware blocks and have XPS re-generate netlists for all the cores that we were using. This was extremely time consuming since the other cores did not change at all. All we needed was to update the netlist for the wrapper. There should be an option to selectively synthesize the core that you want.

2.5 What was Done to Ensure Success

2.5.1 Risk Mitigation

This project was built upon last year's AC97 driver project. With that, we were certain that sound recording and playback was not an issue. However the two biggest unknown was whether or not we could get

a) The hardware audio cores and
b) The Compact Flash system to work.

It turned out that we were not able to get the FIR filter to work. It was not responding to input stimulus when it was programmed into the FPGA. As a backup to the first problem, I suggested we do processing on software. We implemented the variable playback rates as a result. Also we wanted to show that we knew how to integrate a core to the FSL. Hence an echo core was added, whose complexity is not as complex the FIR filter. Details on these processing modules can be found in the group report.

To mitigate the second problem, the ZBT RAM was implemented first to store recorded audio. With this, there was a way to store the recorded audio data even without the Compact Flash. Fortunately we were successful in implementing the Compact Flash for data storage, so the ZBT RAM is used to buffer audio data instead of data storage.

2.5.2 Design Flow and Software Simulation

The design methodology used was defined at the beginning of the course. The general guideline is as follows:

1. Define the problem and what we need to do
2. Break down this problem into smaller, more manageable problems
3. Integrate design blocks into the final system

During the design phase of the hardware components, simulations were performed in ModelSim to verify that the blocks behave as expected. However, because of time constraints, only the behavioral simulations were done. For example, post-synthesis simulation was not performed for the FIR filter core because of the lack of time. Perhaps if the post-synthesis simulation was performed, we may have gotten a better understand of why the core was not responding to our inputs when programmed into hardware.

Because there were only two members in the team, we did not use any advanced version control software such as CVS during the project. We simply emailed ourselves the updated files and kept the old ones in our email archives in case we needed to go back to it. It was an efficient way of collaborating and sharing data given that it was only a group of two.

2.6 Which Modules Were Written by Me

I mainly worked on implementing the FIR filter function for the Xilinx Board. It involved more researching and simulation than actual coding. MatLab was used to simulate the desired frequency response. CoreGen generated the HDL file that represents the core.

The modules that I had to write are the following:

1. Wrapper file (VHDL) to integrate and test the core and the FSL
2. Wrapper file (VHDL) to test the core alone
3. Wrapper file (VHDL) to necessary associated files to package the core to be used XPS
4. VHDL test bench to test each component
5. TCL do script to run Modelsim

The behavior of the modules was tested using ModelSim (described in the Group Report). During the system integration process, I worked with Chris test the C-code for MicroBlaze software.

The wrapper file in (1) was written to include the FSL Models so that when simulated, the external inputs represented inputs from the MicroBlaze. Outputs from this wrapper module represent inputs to the MicroBlaze.

The wrapper file in (2) tests only the core alone. The input to the wrapper imitates data coming out of the FSL from the MicroBlaze processor. The outputs of the wrapper represent inputs into the FSL that directs data to the MicroBlaze.

The VHDL test bench and TCL do scripts were modified so that certain signals can be viewed and more customized and complex waveforms be sent to the core for testing.

2.7 What I Learned

I learned many new things over the last 3 months, especially on topics regarding the design embedded systems and interaction of hardware and software. When programming at a much higher level, all the work is done by the drivers and therefore don't really appreciate the complexity of such systems. For instance writing an audio filter and having it work with the MicroBlaze could be a project on its own. There are so many things, such as timing and memory usage that a systems designer needs to worry about.

What I also about digital logic design is that even it a block of logic works in simulation it might not work on hardware due to a variety of reasons. The main reason could be due to timing requirements were not met or the logic was not synthesized properly.

2.8 Community Contribution

As a group we figured out how to read/write data to the compact flash. I believe that this is a major accomplishment since no other group has gotten this piece of hardware to work. A step-by-step guide on how to setup a base XPS system and code to read and write to the Compact Flash card through the System ACE interface can be found in Appendix I and Appendix II of the group report.

3. Feedback to Xilinx

I believe that Xilinx has excellent products and development tools for its customers. The interface is generally very intuitive and user-friendly. The amount of documentation on the site is overwhelming and I understand that it may be difficult to organize. But for the benefit of all customers, both commercial and academic, better organization of the documents can be done.

Throughout the project, I had the most difficult time finding the information I need. There was simply so much of it to go through, it felt like finding a needle in a haystack. For example, the marketing papers on the System ACE site claims that data could be saved into the System ACE Compact Flash card. How can this be accomplished? Is there an application note that documents this? We managed to figure it out from sifting through numerous documents related to the System ACE interface.

During the course of this project, I did most of the work in the Solaris environment. However the version of ISE for Solaris does not support graphical testbench waveform generation. I think this is a valuable tool to have and should be included in the Solaris version of ISE.

Already mentioned in section 2.4.3, XPS should give the option to selectively synthesize user-defined cores instead of re-synthesizing everything. This would save a lot compilation/synthesis time since 90% us only modify one core at a time.

4. Course Feedback

I enjoyed this course immensely because it was a very practical and hands on course and has allowed me to apply what I have learned in previous years. It was even more appealing to me because of the fact that it has no final exam.

It would be better if the project started a bit earlier. I felt that we spent too much time running through the labs. The lab demo was not worth very much yet we spent a month on it. To mitigate this, I would suggest that each group of students sign out a Xilinx board (like what we did during the we prior to the Project Demos). This goes hand in hand with the open lab concept. This way we would be able to go through the tutorial modules at our own pace, which I believe is very similar to how things work in industry.

During the lab hours, it would be good to have Both TA's present for assistance. Additionally, we should have weekly 1-on-1 with the TA or with the instructor to get feedback on the progress of the project. This way, we would get guidance and not underestimate the complexity of a certain goal.

In terms of the grading structure, I think that the individual report is weighted too much. I believe that the group report should be weighted more, since this documents the contributions of the entire group, which also reflects the co-operation of each team member. Secondly, more marks should be given for the project Demo because it represents a significant amount of work to get the demo working.

5. Final Word

This course although difficult at times is very rewarding. I certainly did make the right choice of registering for this class, as I have learned a lot and have something to write about in my resume. I would like to thank Professor Chow for offering this course and TAs Chris and Lesley and everyone else who provided me with valuable advice and assistance along the way.