EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

Abstract

Many audio effects are very popular among artists to be used in live concerts. However, latency problems may occur when using these effects. This latency may affect the artists negatively or cause the effect not to be used well. This project aims to minimize the latency of a audio effect by designing a specific hardware architecture on FPGA as an accelerator. The audio effect accelerator will then be connected to an ARM processor core within a Zynq-based programmable SoC platform. The ARM processor core may be responsible to handle data movement operations, enable or disable the audio effect over captured audio signal and performs some of the operations of the audio effect algorithm. The audio effect algorithm will first be written on MATLAB & Simulink and corresponding hardware is generated on MATLAB with HDL Coder Transform tool. Our main objective in describing the hardware of the audio effect algorithm in is to reduce the latency of audio signals from input to output. Thus, various experiments will be performed to find the lowest possible latency solution for the selected audio effect. The System-on-chip solution will be prototyped in ZedBoard that contains ZC7020 Zynq chip. Processes on audio signals cause latency in processors which depend on the speed of the processor. To reduce this latency, it is aimed to design an application-specific hardware accelerator that will be implemented on the FPGA logic of the Zynq chip. The processing system of the Zynq device will manage the accelerator.

After the project is realized, a decrease in latency is expected and the solution can be used in areas such as live music concerts.

1. Introduction

The music industry is an industry with a huge economy and its economy is growing every year. From IFPI Global Music Report 2019 [1], the global recorded music market grew by 9.7% in 2018. IFPI's Global Music Report 2019 show total revenues for 2018 were $19.1 billion. Certainly, some sound effects are needed while producing these many songs. Therefore, audio effect is one of the critical tools of today's music industry. Any latency that occurs when applying sound effect can cause trouble. In addition to the successful operation of audio effects, the delay should also be minimal. LowLAG is being developed so that users can implement audio effects applications with minimum latency.

What does latency mean in audios? Latency refers to a short period of delay (usually measured in milliseconds) between when an audio signal enters a system and when it emerges. The shorter this period of delay, the less the latency will be. Minimizing the latency will allow us to prevent the loss of time in audio effects. The human ear can not to detect below 20 milliseconds latency [2]. The main objective is to lessen the latency to times that are too short for the human ear to detect. First of all, it is necessary to identify the factors causing the delay in order to minimize the latency. Possible reasons for the latency are as follows: Analog to digital conversion, buffering, digital signal processing, transmission time, digital to analog conversion. Many implementation techniques such as filters, delay lines, time-segments, and time-frequency representations which are used for audio effects are related with the digital signal processing because of that digital signal processing is one the indispensable application in the digital real time audio effect implementation [3]. We will focus on speeding up latency in the digital signal processing application.

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

Until a few decades ago, digital signal processing was implemented either on the hardware design or in the software design. Both designs have their own advantages. We can obtain low latency and high reliability applications with the digital signal processing on only hardware design, but this gives us very little flexibility and more complexity compared to the design in the software [4]. We can obtain the more flexibility with the software design. Therefore, it would be more effective to implement hardware and software together on a chip, this implementation is called System on Chip. There are many studies in the field of application of digital audio effect on Soc system [5][6]. We will use ZedBoard Zynq-7000 EPP Development kit for SoC implementation [7]. ZedBoard provides us the opportunity to develop both for the software part and for the hardware part together.

2. Method

LowLAG aims to decrease latency (which is the passing time between audio input to audio output) that caused by the load on the CPU while applying the sound effect. System on Chip design will be used to decrease that CPU load. It is desired to speed up the sound processing operation with FPGA because of latency with software solutions and to continue to benefit from the software for functional settings.

Sound effect algorithms were made on MATLAB & Simulink. While creating sound effect algorithms on MATLAB & Simulink, some components in MATLAB & Simulink's library were used. In this way, an algorithm design was created that would make a sound effect. This design will be further developed and some components like delays, buffers, some filters will be added to increase the variety of sound effects. With the development of the design, creating more quality sound effects are planned. This design, which was made thanks to the advantage of MATLAB & Simulink program, was reduced to the HDL code with its own tool. Through this transformed design, the IP to be used for the sound effect was created. And this IP was added to our design as sound effect thread.
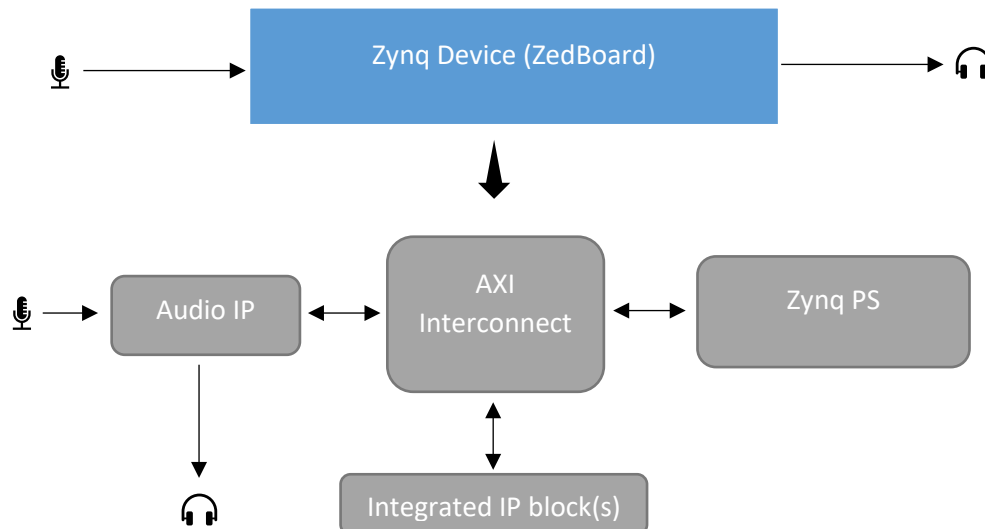

Figure 1: How audio system will work on LowLAG

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

As a result, the system we created works as follows. The microphone that we connect to ZedBoard's audio input port, gets sound from outside. The received sound is controlled via the Zynq processor, the processor of the ZedBoard. The Zynq processor controls the application of the effect we designed on the IP to the incoming sound by communicating with the sound effect IP with the necessary connections with the processor. Then, it gives the effected sound out through the speaker that connected to the audio output port on it.

2.1. Using of Zynq Device for This Project

Purpose of the using Zynq device is decrease the latency as mentioned above. In this case, solution must give high performance, also not lose ease of use and also shouldn't be too expensive for marketability. Application specific integrated circuits (ASIC) gives more performance on applications and less flexibility compared to software and creates too much costs. FPGA (Field Programmable Gate Arrays) makes things easier. It gives more performance than software and less costs than ASIC chips. [8] On the other hand, a system-on-chip solution gives performance of FPGA and flexibility of software. This is why LowLAG is a SoC project.

It is aimed to speed up the system by making the sound processing part on the programmable logic. The thought is converting the MATLAB & Simulink design of the sound effect (software part) to IP block via HDL Coder Transform tool and use with Zynq device. All communications between PS and PL are provided by AXI Interconnect. IP block that is used for sound processing will use AXI4_Lite interface. The prepackaged I2S controller for the Analog Devices ADAU1761 audio codec on the ZedBoard will used for transmitting input and output signals. Objective working principle is shown on Figure 1 roughly.

LowLAG consists of two main part which are hardware and software as mentioned before. These two parts are discussed below.

2.2. Hardware

Hardware will be main part of the processing operation. Use hardware for processing operations are more efficient than software solutions.

Hardware has been designed for a simple sound effect application. The design was created to make changes to the incoming sound and transmit it to an output. Hardware design was made on MATLAB & Simulink. When choosing this tool, the ease of converting the design to HDL code was taken into consideration. The design has been converted to IP that will work on ZedBoard.

2.2.2. Creating IP

In this step, desired work is creating an IP block to use for sound processing. The MATLAB & Simulink design will be used for creating that IP block using by HDL Coder Transform tool. Research process is still going on about that subject.

While creating the design, components supporting MATLAB & Simulink's HDL Coder were used. Necessary tests were first performed on MATLAB & Simulink. It was seen that the effect was successfully applied in

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

the tests of the created design on the MATLAB & Simulink. And then it was converted into an IP block to be used in our system with the HDL Coder Transform tool.

2.2.3. Using Audio on ZedBoard

Another IP block is going to used for using audio ports on ZedBoard. This IP block take input signal and communicate with other IP block which will process the sound and PS, after it will give the final processed signal to the output. This block uses I2S interface standard and ADAU1761 audio codec on ZedBoard.

2.3. Software

The aim of using software is protect the flexibility of the effect that had on software solution before. The software part will be worked with hardware together. Some customization will be used in software part like settings etc.

2.3.1 Application

Application part will provide an access to changeable features in sound effect. For example, consider that a reverb effect is running on LowLAG. In the software part of LowLAG you will change the feature such as depth of sound, size of room, duration of the effect or mix of sound. These are some of the examples and these can be changed depending on sound effect. The main idea in there is not to lose the flexibility on the effect while the acceleration process happens. The application part will provide that.

3. Empirical Setup

ZedBoard is used for realizing the project as mentioned on the previous section. Test applications were made to get audio from ZedBoard and Vivado 2015.1 was used. The origin of experiment that we are worked on was taken from Zynq Book Tutorial which is described below.

3.1. Design of Zynq Book Tutorial Exercise 5

This is the main section of the project now working on. It is based on Zynq Book Tutorial Exercise 5. Its programmable logic part consists I2S audio controller and IP blocks that provide effect different than mostly used blocks. These IP blocks are available on present book sources.

PL and PS part will be worked together. Firstly, it is ensured to giving input and taking output correctly. The sound effects that are wanted to place in PL is converting an IP block via Vivado HLS and MATLAB HDL Coder. Also, software part was reorganized in order to that effect. Figure 2 describes the system briefly. These sections have not so much details. But details will occur as the project progresses.

There were 2 option to get measurements about project results. Latency could be measured with an oscilloscope from input and output signal. It was an analog way to get results. But we cannot get result with that way because our laboratories are not available for use due to COVID-19. The other option is getting latency values from chip clock. Starting of input signal and starting of output signal can be observed from clock hit and can be convert easily to time in milliseconds. In order to do that, TTC (Triple Timing Counter) time is going to used. On the other hand, the latency without ZedBoard is going to measure with a record in Windows.

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
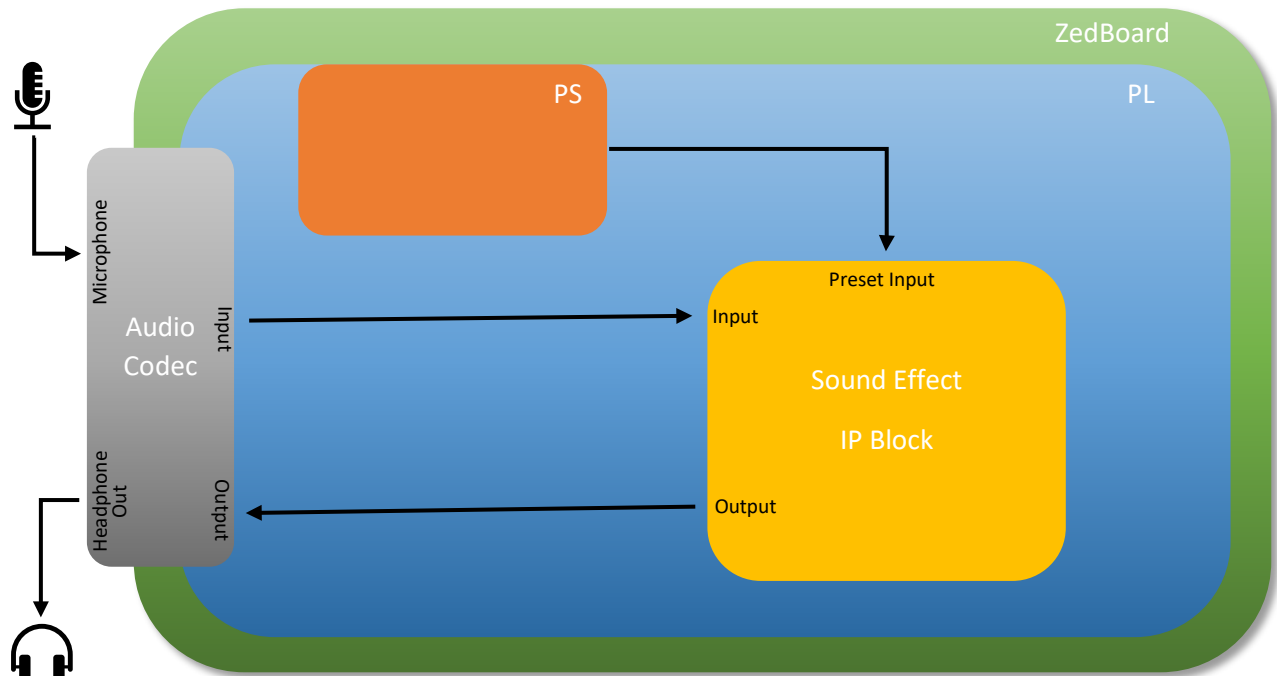Team Name: Double Y
There is no link available for now.

Figure 2: Overview of complete SoC design

3.2. IP Creation with HDL Coder in MATLAB & Simulink

In this section, a basic sound effect was created in Simulink and converted to an IP block with HDL Coder. The sound effect is a distortion effect and that is also experimental, so that is not good enough to use in music applications. But it gives an idea to how we can improve our system.

3.2.1. Simulink Model of Sound Effect

The design of sound effect is very simple and basic as mentioned before and can be changed for future applications to get more clear results. The point is in this section how did we do that? A design should be created in a Simulink model to create new IP. Figure x shows that basic distortion effect.
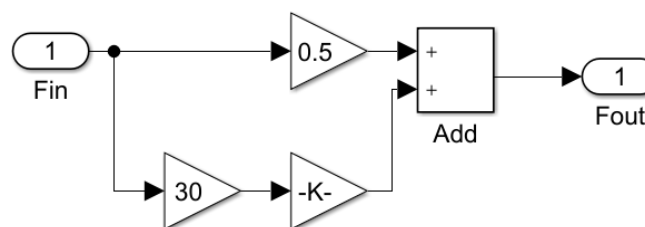

Figure 3: The design of distortion effect.

The main idea of that effect is distorting the audio and mix with the clean audio. The packaged subsystem (shown as Figure 4) uses fixed point variables to compatible with the ZedBoard.

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
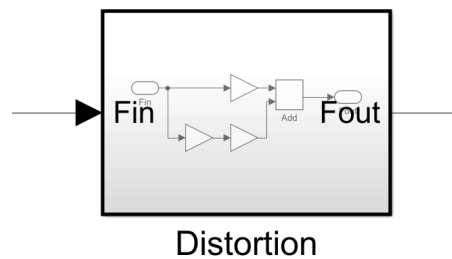There is no link available for now.

Figure 4: Subsystem of distortion

MATLAB needs an HDL tool for creating HDL codes. The HDL tool must be setup in MATLAB project's folder for that reason. In order to do that, the code in below must be entered in command window of MATLAB. It is used Xilinx ISE for HDL code generation and tool path shows where Xilinx ISE's location is in the PC that we made the experiment.

```
hdlsetuptoolpath('ToolName', 'Xilinx ISE', 'ToolPath',
'D:\Programlar\Xilinx\14.7\ISE_DS\ISE\bin\nt\ise.exe')
```

The HDL Coder can be used after that stages. To generate HDL code, right click on distortion block and enter HDL Workflow Advisor. A new window opens that shows what must be done before the generate HDL code. Input parameters must be entered shown as in Figure 5. After that, Coprocessing – blocking should be selected as the processor/FPGA synchronization. AXI4-Lite is selected automatically as platform interface. Also addresses of input and output ports can be selected at this stage. Our effect uses x"100" as input and x"104" as output, which names are Fin and Fout, respectively. HDL code is created as VHDL and there is no pipeline process for now to get better latency results. We will focus on that in next stages. The HDL code can be generated if all tasks have run without an error.

Figure 5: Set Target Device and Target Tool

3.2.2. IP Packaging

IP packaging process has done in Vivado 2015.1. The changes in the packaging step are related with ports and interfaces, and memory. Vivado provides Auto Infer Interface chooser. AMBA AXI Interface (aximm) was selected in that stage in order to create IP compatible with AXI Interface. In the memory part one register has been used and range of that register was changed to 32. Packaging process has finished when click the Package IP button on review and package section.

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.
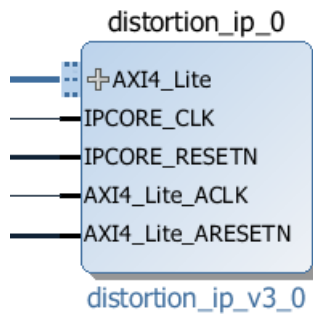
3.2.3. Adding IP to the Design

Figure 6: IP Block of Distortion Effect

An IP repository must be added to Vivado before adding the IP. Any IP can be added in selected repository. These processes are made in IP Catalog Settings. These repository and IP folders were already created in IP packaging process. After adding IP from IP repository, IP can be viewed in UserIP folder in IP Catalog. Now, our custom sound effect IP can be implemented to our Zynq device design. We added the Distortion IP to our design as shown in Figure 7. Designer Assistance is available for make connections, but two connections must be done manually which can be shown in Figure 7. When Distortion IP added to design first time, IPCORE_CLK and IPCORE_RESETN connections were not done because of we did not define in the packaging. Therefore, IPCORE_CLK must be connected to AXI4_Lite_ACLK and IPCORE_RESETN must be connected to AXI4_Lite_RESETN. Finally, our custom IP is ready to use. We can generate bitstream in this stage. Last thing that we need to do is edit software part for use.
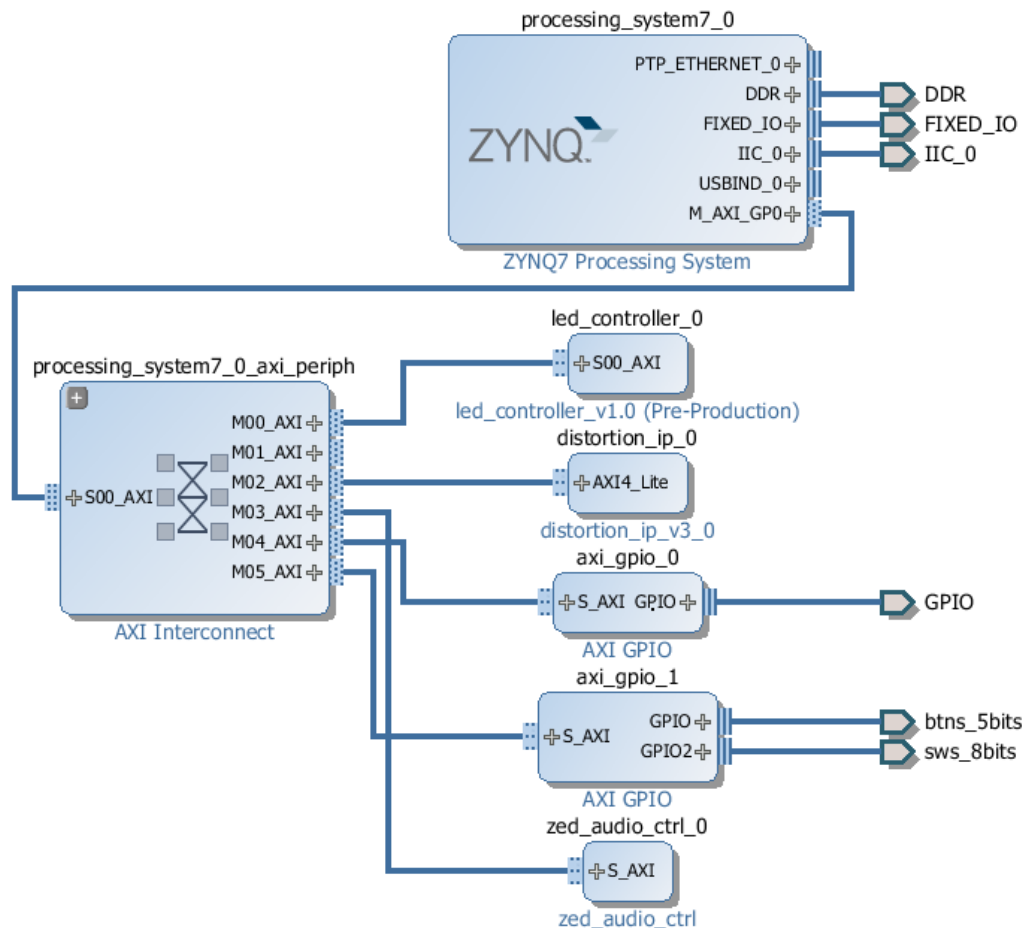

Figure 7: Design of LowLAG with Distortion IP Block

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

3.3. Prepare Software in SDK

When bitstream is generated that must be exported to SDK include bitstream. New empty application project must be created with new board support package and new repositories must be added to project in Tools > Repositories window. New custom IP's can be shown in the BSP's Settings on system.mss file.

Driver files (that named distortion_addr.h and audio.h) must be imported to design to use custom IP and audio blocks. We wrote the distortion_addr.h like below as we designed in the HDL Coder part. These addresses can be found in the HDL Coder's summary report.

---

distortion_addr.h

This file is changed for the distortion IP and its addresses that is used.

```
#define  IPCore_Reset_distortion     0x0  //write 0x1 to bit 0 to reset IP core
#define  IPCore_Enable_distortion    0x4  //enabled (by default) when bit 0 is 0x1
#define  IPCore_Strobe_distortion    0x8  //write 1 to bit 0 after write all input data
#define  IPCore_Ready_distortion     0xC//wait until bit 0 is 1 before read output data
#define  IPCore_Timestamp_distortion 0x10
#define  Fin__Data_distortion        0x100  //data register for port Fin
#define  Fout__Data_distortion       0x104  //data register for port Fout
```

Also, source files of application (adventures_with_ip.c, adventures_with_ip.h, audio.c, ip_functions.c) imported into the project from source files of Zynq Book Tutorial.

The changes we made in the codes can be shown below.

---

adventures_with_ip.c

These changes were made for UART console information.

```
xil_printf("Embedded Distortion Demo\r\n");
xil_printf("Enter 's' to stream pure audio, 'n' to add tonal noise, 'f' to distortion
when pressed and 'd' to distortion continuous.\r\n");

case 'f':
      xil_printf("ENTERING DISTORTION OPERATION\r\n");
      xil_printf("Press 'q' to return to the main menu\r\n");
      distortion();
      break;
```

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

---

adventures_with_ip.h

These definitions give us more readable codes. It takes the address values from distortion_addr.h and xparamaters.h header files.

```c
#include "distortion_addr.h"

void distortion();

#define DIST_LOC XPAR_DISTORTION_IP_0_BASEADDR
#define DIST_FIN DIST_LOC + Fin__Data_distortion
#define DIST_FOUT DIST_LOC + Fout__Data_distortion
#define DIST_STROBE DIST_LOC + IPCore_Strobe_distortion
```

---

ip_functions.c

This section is the main difference for the software part. That makes the connection between PS and PL parts that is necessary for using Distortion IP.

```c
/* -------------------------------------------------------------------------- *
 *                              distortion()                                  *
 * -------------------------------------------------------------------------- *
 * This function performs distortion effect. It takes the audio from input and
 * gives an output that is mix of original audio and distorted audio together
 * when any button is pressed, otherwise the original audio will be streamed.
 * These processes are done in PL part with created custom IP named Distortion
 * IP. The input and output audio are provided by audio codec.
 *
 * The main menu can be accessed by entering 'q' on the keyboard.
 * -------------------------------------------------------------------------- */
void distortion(){
      u32  in_left, in_right, out_left, out_right, prevL, prevR;

      while (!XUartPs_IsReceiveData(UART_BASEADDR)){
            // Read audio input from codec
            in_left = Xil_In32(I2S_DATA_RX_L_REG);
            in_right = Xil_In32(I2S_DATA_RX_R_REG);

            /* ----------------------------------------------------------------- *
             * ------------------------ LEFT CHANNEL ------------------------- *
             * ----------------------------------------------------------------- */
            if(in_left != prevL) /* New left sample? */
            {
                  /* Define the signal in new variable. */
                  out_left = in_left;
```

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

```c
            Xil_Out32(DIST_FIN, out_left >> SCALE); // Input of distortion
            Xil_Out32(DIST_STROBE, 0x01);          // Stobe distortion to
                                                   // signal inputs are finished

            /* If any button is pressed */
            if(XGpio_DiscreteRead(&Gpio, BUTTON_CHANNEL)>0){

                    /* Wait until output data is ready */
                    out_left = (Xil_In32(DIST_FOUT) << (SCALE-1));
                    // Output distorted audio
            }

            /* Output audio to the codec */
            Xil_Out32(I2S_DATA_TX_L_REG, out_left);
        }

        /* ------------------------------------------------------------------- *
         * ------------------------- RIGHT CHANNEL ------------------------- *
         * ------------------------------------------------------------------- */
        if(in_right != prevR) /* New right sample? */
        {
                /* Define the signal in new variable. */
                out_right = in_right;

                Xil_Out32(DIST_FIN, out_right >> SCALE);// Input of distortion
                Xil_Out32(DIST_STROBE, 0x01);          // Stobe distortion to
                                                       // signal inputs are finished

                /* If any button is pressed */
                if(XGpio_DiscreteRead(&Gpio, BUTTON_CHANNEL)>0){
                        out_right = (Xil_In32(DIST_FOUT) << (SCALE-1));
                        // output filtered audio
                }

                /* Output audio to the codec */
                Xil_Out32(I2S_DATA_TX_R_REG, out_right);
        }

        /* Update previous input values */
        prevL = in_left;
        prevR = in_right;

        // Write audio output to codec
        Xil_Out32(I2S_DATA_TX_L_REG, in_left);
        Xil_Out32(I2S_DATA_TX_R_REG, in_right);
    }

    /* If input from the terminal is 'q', then return to menu.
     * Else, continue streaming. */
    if(XUartPs_ReadReg(UART_BASEADDR, XUARTPS_FIFO_OFFSET) == 'q') menu();
    else distortion();
} // distortion()
```

3.4. Physical Setup

In that section, PL and PS parts are ready to go. ZedBoard can be connect to the PC with UART and JTAG. Furthermore, 3.5 mm audio jacks must be plugged. The input port that we used is line in and output port that we used is headphone. The application can be used now.

4. Empirical Results

As a result, a design was created for the application. This created design is designed to apply distortion effect to the incoming sound. As a result of this design, an application that gives the effect applied sound is created and tried. As a result of the application, it was checked whether the distortion effect was applied successfully and the length of time between the sound input and output which is called latency the during application. Various observations were made in order to minimize the delay for sound effect applications, which is the main purpose of this project. Successful application of sound and delay, which is the main criterion for the success of the project, has been observed successfully. While making these observations, the following steps were taken.

The created sound effect design was first run on MATLAB & Simulink. It was observed that effects were successfully applied to the input sound. However, in this experiment, it was seen that there was a delay which noticeable by ear between the sound input and the output of the effect sound. The magnitude of this delay was at a level that would cause problems in terms of implementation. The amount of delay, which is the difference between the input and output sound, was observed as follows. While the sound effect application was working on MATLAB & Simulink, the input and output sound was analyzed in detail during the time domain with DirectSound as input and output driver. The results obtained were noted in the Table 1 and spectrum of one of the recorded audio file is shown below in Figure 8.
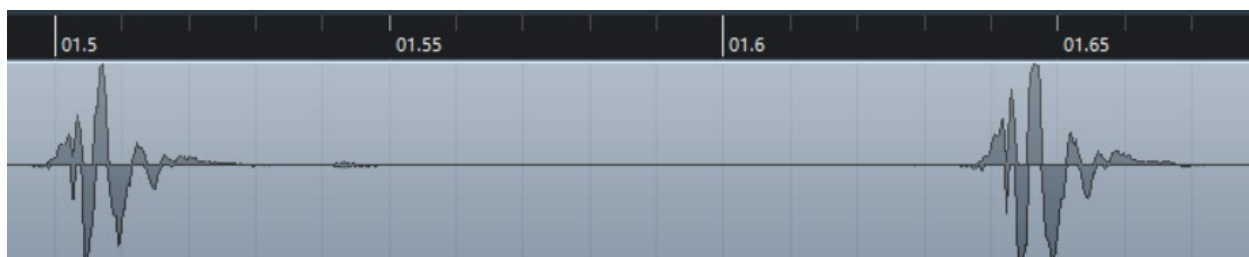

Figure 8: Input and Output in a Single WAV file for Distortion IP Simulink Model.

Input Time of Sound	Output Time of Sound	Samples Per Frame	Buffer	Latency(ms)
04.852	05.057	1024	-	205
01.502	1.641	128	128	139
01.989	02.189	1024	128	200

Table 1: Latency on MATLAB & Simulink Environment with Various Conditions

To minimize this delay, the application was tried to run on other platforms.

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

For this reason, our application has been transformed to run on Zynq device, System on Chip application card, which has the opportunity to design hardware and software together to speed up the sound effect design we designed. As a result of the trials, it was observed that the effect was successfully applied in the distortion output sound. Another criterion which is minimum delay was observed to be too low to be noticed by the ear. A detailed measurement was made to calculate the delay with the TTC Timer module. First of all, TTC timer is started before the beginning of distortion() function and finished end of the distortion() function. The codes and result are shown as below in Figure 9.

```
XTime_GetTime(&tStart);
distortion();
XTime_GetTime(&tEnd);
xil_printf("Output took %llu clock cycles.\n", 2*(tEnd - tStart));
ElapsedTime = 1.0 * (tEnd - tStart) / (COUNTS_PER_SECOND);
xil_printf(" Output took %.9f s.\n",ElapsedTime);
```

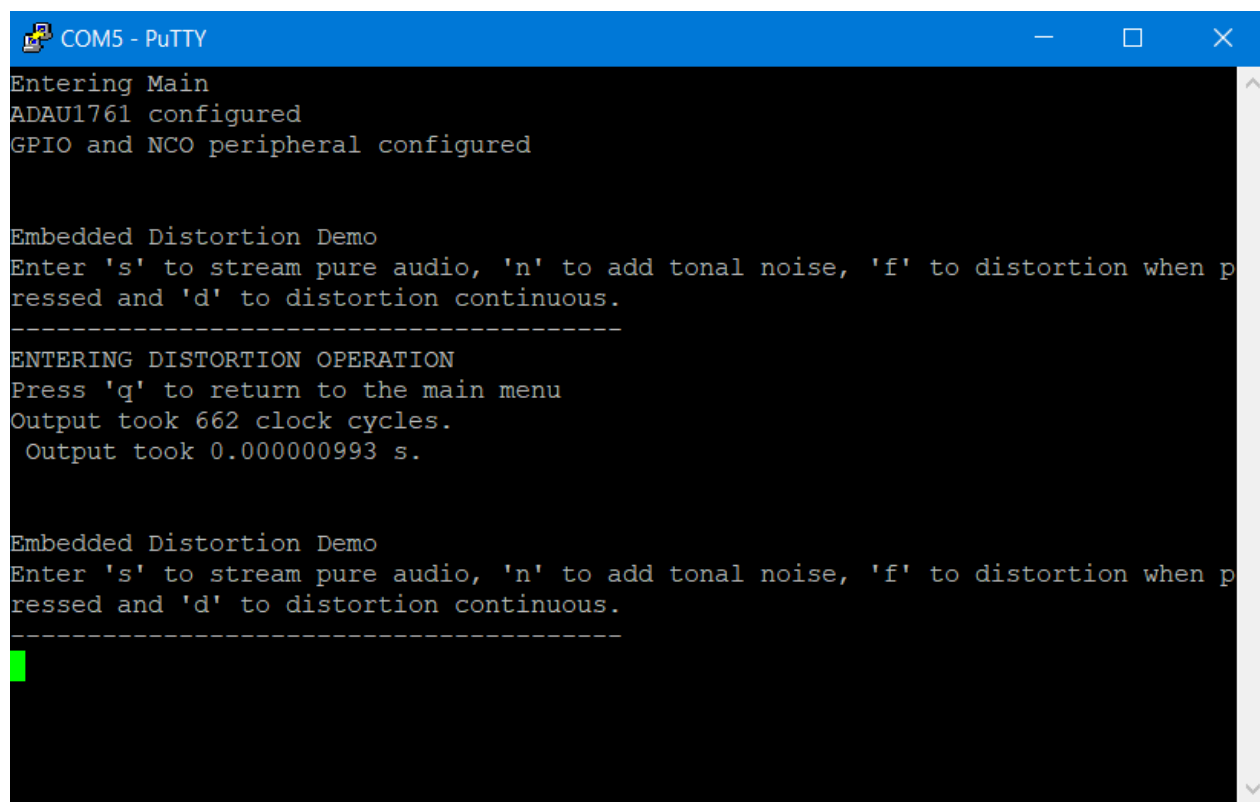*(Thanks to İsmail SAN for helping us for measurement of latency in the chip.)*

Figure 9: Output of UART connection and elapsed time results for distortion() function.

EEM 464 – System on Chip Design: Project Progress Report 1

Yunus Emre Esen
Yunus Kalkan
Team Name: Double Y
There is no link available for now.

5. Conclusion

Our main goal in the LowLAG project is to minimize the latency in sound effect applications. We know that sound effect applications are used by many users today in many areas. One of the biggest criteria for tool selection in users is the latency during implementation. In our project, we will use the Zynq 7000 ZedBoard, which has the opportunity to make hardware/software co-design on SoC to minimize these latencies. LowLAG guarantee users a smooth experience in sound effect applications. With the LowLAG, it is aimed that the delay in sound effect applications will be too low to be noticed by the human ear. In this way, it is our expectation in this project to have a sound effect application that will be preferred by many users in the music industry, which is very wide. In this way, it is our expectation the LowLAG to have a sound effect application that will be preferred by many users in the music industry, which is very wide.

The results showed us*, **LowLAG decreased the latency for Distortion IP <u>from 139 milliseconds to 0.993 milliseconds</u>.*** 139.98 times speed up has provided on audio latency with LowLAG on ZedBoard. We should repeat that the driver was DirectSound in the Simulink model.

Other IP blocks will be tried for other effects. Also, pipelined versions which provide getting less latency will be implemented on to design in the near future.

References

[1] IFPI Global Music Report 2019, https://www.ifpi.org/news/IFPI-GLOBAL-MUSIC-REPORT-2019

[2] Juillerat, Mueller, Schubiger, REAL-TIME, LOW LATENCY AUDIO PROCESSING IN JAVA, Proceeding of the International Computer Music Conference, 2007.

[3] Udo Zölzer, DAFX: Digital Audio Effects Udo Zölzer, Library of Congress Cataloguing-in-Publication Data,Second Edition e-PDF ISBN 978-1-119-99130-4

[4] Uwe Meyer-Bäse, Digital Signal Processing with Field Programmable Gate Arrays, Signals and Communication Technology. Springer-Verlag, Berlin, Hei-delberg, NewYork, 2 edition, 2007, ISBN-13 978-3540211198

[5] Markus Pfaff, David Malzner, Johannes Seifert, Johannes Traxler, Horst Weber, Gerhard Wiend,IMPLEMENTING DIGITAL AUDIO EFFECTS USING A HARDWARE/SOFTWARECO-DESIGN APPROACH, FH-OÖ/Hagenberg, Dept. HSSESoftwarepark 11, A-4232 Hagenberg/Austria

[6] Kyungjin Byun, Young-Su Kwon, Seongmo Park, and Nak-Woong Eum, Digital Audio Effect System-on-a-Chip Based on Embedded DSP Core

[7] www.zedboard.org

[8] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart, The Zynq Book, 1st Edition