EE296A Embedded SoC Design Spring 2017

San Jose State University
Department of Electrical Engineering
Laboratory Assignment #3

Question: You will design a 58-tap FIR filter in Xilinx SDSoC using C/C++ and accelerate it on the hardware using Zybo Zynq board.

NOTE: All files should be extracted to locations without any spaces in their names.

Before designing the filter, a base platform needs to be setup for SDSoC. To do this follow these steps.
a) Extract the provided zybo_fir.zip file to a location of your choice.
b) Start Xilinx SDSoC 2016.2
c) When SDSoC starts change the workspace to the location of your choice. This will be your project directory.
d) Click on the *Create SDSoC Project* in the Welcome tab or select **File > New > SDSoC Project.**
e) Enter *Project name.* This will not be referred as *<Project Name>*
f) For the *Platform,* click on the **Other…** button, browse to the location where you extracted zybo_fir.zip and select the zybo_fir extracted folder.
g) For the *OS*, select **Standalone**, and Click **Next**.
The Templates window will be displayed with Audio Playback as one of the two possible templates.
h) Select the *Audio Playback* application and click **Finish**.
i) Right-click on *<Project Name> in the Project Explorer window* and select **Build Project**.

Connect the ZYBO board to your computer using a micro-usb to usb cable and connect an audio patch cable between the PC's headphone output and Line-In connector of the board. Connect a headphone to the HPH OUT connector on Zybo board. Connect the board and power it ON. Right-click on the *<Project Name>* folder and select **Run As > Launch on Hardware (SDDebug)** to run the application. Play some music on the PC and you should be able to hear the same on your headphone. **Note** that the audio is noisy, the FIR filter that you will design will filter some of the noise, and improve the sound quality.

To design your filter, extract the contents of filter_src.zip and start a new SDSoC Project
a) Select a new <*Project Name*>
EE296A/EE277A Embedded SoC Design Spring 2017
Y. Kim (9/23/2015)
b) For the *Platform,* click on the **Other…** button, browse to the location where you extracted zybo_fir.zip and select the zybo_fir extracted folder.
c) For the *OS*, select **Standalone**, and Click **Next**.
d) For the *Template*, select empty project, and Click **Finish.**
e) Right click the *src* folder in the Project Explorer window and select Import.
f) Choose *General->Filesystem*, then find the folder you extracted from filter_src.zip and select that folder.
g) Check all the boxes of the folder's contents and click **Finish**.
a. Folder's contents should include: audio.c, audio.h, fir_coef.dat, and fir_main.c

h) Now edit the fir_main.c code to implement your fir filter. You can insert your filter replacing the commented line: *//Implement Filter here*
i) After building double click on the project.sdsoc file in the file explorer
j) In the *Project Overview* window, Click on the + symbol next to Hardware Functions.
k) In the pop-up selected fir(data_t *, data_t) and select **OK**
l) When finished, build the project. This will take a while.
m) When the build is complete, copy the contents from <*Project Name*>\SDDebug\sd_card into the SD card.
n) Test out your design and see if you hear any noise or static. If you don't you are done. If not, you must work on it more.

Solution:

FIR CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <xil_io.h>

#include "xiicps.h"

#include <xparameters.h>

#include "xuartps.h"

#include "audio.h"

#include <sds_lib.h>


unsigned char IicConfig(unsigned int DeviceIdPS);

void AudioWriteToReg(unsigned char u8RegAddr, short u16Data);

void LineinLineoutConfig(void);

void audio_sample_write(int DataL, int DataR);

void audio_sample_read(int * DataL, int * DataR);

void audio_sample_wait(void);


XIicPs Iic;


#define N       58

#define M   59                          // defining variable M for 1st while loop //


typedef short   coef_t;

typedef short   data_t;
typedef long     acc_t;


void fir (

            data_t *y,

            data_t x

            ) {

            const data_t c[N+1]={

           #include "fir_coef.dat"

                     };


            int i, total;                  // defining int i for 2nd while loop and total to store
answer of convolution each time //

            total = 0;                     // make total zero to erase the starting garbage value //


            float j[M];                    // defining float j with array of 59 to store the value of x in
it //


            i = 0;                         // to start loop from 1 make i zero //

            while (i < M)                  // start of while loop //

            {

                      j[i] = 0.0;          // make j array zero to erase the starting garbage value //

                      i++;                           // increament i by 1 //

            }


            i = 0;                         // again make i zero for next loop operation //

            while (i <= N)          // start of while loop for convolution or FIR operation //

            {

                     j[0] = x;             // transfering value of x input to j[0] //

                     total += j[i] * c[i]; // equation for convolution operation and saving result to total
for 58 times //
```

```
                if(i > 0)                      // loop of if to save the value of j after each convolution
operation //
                {
                        j[i] = j[i - 1]; // equation to save the previouse value //
                }

                i++;                                    // increament i by 1 //
         }

        *y = total >> 16;                      // shifting the final convolution answer by 16 digit to store
it in short *y ouput //
        // *y = x;
         //Implement Filter here
        }



#define SAMPLES N+5
void filter(void)
{
        int i=0;
        data_t signal, output;

        int DataL, DataR;
        int FilOutDataL, FilOutDataR;

        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR, 0b1);
        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR + 0x4, 0b1);
  for (i=0;i<SAMPLES;i++) {
         if(i==0)
                 signal = 0x8000;
         else
                 signal = 0;
```

```
            fir(&output,signal);
            printf("%i %d %d\n",i,(int)signal,(int)output);
    }

        while (1)
        {
                audio_sample_wait();
                audio_sample_read(&DataL, &DataR);
                fir(&FilOutDataL, (DataL>>8));
                fir(&FilOutDataR, (DataR>>8));
                FilOutDataL = (FilOutDataL & 0x0000ffff) << 8;
                FilOutDataR = (FilOutDataR & 0x0000ffff) << 8;
                audio_sample_write(FilOutDataL, FilOutDataR);
        }
}

int main(void)
{

        //Configure the IIC data structure
        IicConfig(XPAR_XIICPS_0_DEVICE_ID);

        //Configure the Line in and Line out ports.
        LineinLineoutConfig();

        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR, 0b1);
        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR + 0x4, 0b1);

        filter();

}
```