ISOLATED SPOKEN WORD RECOGNITION USING BF561

By:
Muhammad Saqib Rabbani 1199335
Khaled Jaman Algahtani 1043214
# Contents

## Section #1: Khaled Algahtani 1043214

1. Introduction ...................................................................................................................... 2  
2. ADSP BF561 kit connections ....................................................................................... 2  
3. Front-end analysis ........................................................................................................ 3  
   3.1 Introduction .............................................................................................................. 3  
   3.2 Process description ................................................................................................. 3  
   3.3 Training and Testing stages .................................................................................... 4  
   3.4 Memory allocation .................................................................................................. 5  
3.5 Samples of output results .......................................................................................... 5  
4. Direct Memory Access DMA ....................................................................................... 7  
5. Project files .................................................................................................................... 7  

## Section #2: Muhammad Saqib Rabbani 1199335

6. End point detection ....................................................................................................... 8  
   6.1 Method .................................................................................................................... 8  
   6.2 Data Flow ................................................................................................................ 9  
   6.3 Memory Requirement and Allocation ..................................................................... 10  
7. Dynamic Time Warping .............................................................................................. 10  
   7.1 Method .................................................................................................................... 10  
8. TV Display .................................................................................................................... 11  
   8.1 Method .................................................................................................................... 11  
   8.2 Video Codec Configuration ..................................................................................... 12  
9. Performance Analysis and Experimental Results ....................................................... 12  
10. Conclusion ................................................................................................................... 13  
References ....................................................................................................................... 14  

1. Introduction.

There are many algorithms of speech recognition are used. Each algorithm has specific advantages and disadvantages. The aim of this assignment is to create a program performing recognition of isolated spoken words. This program will extract the features of four words and store it as templates. These templates will be used by applying the dynamic time warping (DTW) algorithm to compare the features of a live audio signal. The DTW is commonly used because it is flexible for implementation on hardware and uncomplicated (Jing, 2010). There are many steps must be done to prepare the templates such as sampling, overlapping, applying Fast Fourier Transform FFT and Discrete Cosine Transform DCT which called Front-end analysis of Automatic Speech Recognition ASR. The live signal (unknown word) is passed through the same process and then the DTW algorithm will be applied to decide which word was spoken. In this technique, the spoken word is surrounded by silence or background noise which makes it easy to know the beginning and the end of the word. This technique is used only when a specific word needs to be detected (YUAN, 2004).

The main challenge in this assignment is the implementation of DTW algorithm on ADSP BF561 which was designed to support fixed-point computations with limited resources such as memory. In general, speech recognition algorithm contains of floating points and complex functions which make it undesirable for cheap DSP hardware (YUAN, 2004). However, optimisation is very important to increase the performance of the system and the efficiency of the final results. In implementation process, there are many problems such as memory management and accessing, serial port configuration, core cycles and linker errors. The system is designed to operate as a single core.

2. ADSP BF561 kit connections.

![Figure 2.1: BF561 connections.](image-url)
3. Front-end Analysis.

3.1 Introduction.

Front-end Analysis is the first step before applying DTW (YUAN, 2004). The purpose of this step is to determine the parameters of the signal which will be used as a reference and store it in the memory to use it in the comparison with the unknown signal. Some of exercise examples are used and modified and C codes were created to calculate the features of the four words (one, two, three and four). The system is designed depending on the flow chart as shown in figure 3.1 to prepare the templates (training stage) and then to decide which word was spoken (testing stage). The C codes are existed in the source file named (process data 1.c).

![Figure 3.1: System Flow Chart.](image)

3.2 Process description.

CODEC AD1836A has four ADC channels operate at sampling frequency Fs = 48 kHz (Pycock et al., 2012). Input signal is transferred to processor through Sport0 and the setup of the SPI to configure the AD1836 was done in (Initialize1.c) source file.

![Figure 3.2: Sampling and overlapping (Bourouba et al., 2006).](image)
The built-in circular function buffer is used to transfer input signal to the process buffer and used to fill the input array which is used as an input samples to the Fast Fourier Transform FFT function. The FFT is used to transfer time-domain samples to frequency-domain. The frame length is 1024 samples (VECTOR_SIZE) with overlapping 50% with the adjacent frame as shown in figure 3.2. The built-in function rfft is used to calculate the spectral values. Furthermore, these values are complex containing real and imaginary parts. The twiddle function is used to produce Twiddle coefficients tables which are used with FFT. The twiddle coefficients must be at least half of the frame length and in this case the twiddle coefficients are equal to the frame length. However, the speech recognition system deals with real values and to overcome this problem a magnitude of the complex values are used (YUAN, 2004). Moreover, the next step is calculating log values of the first 100 values of the frame to produce a log-magnitude spectrum. This process is repeated for each frame and the results are stored in 2D array which is called spectrogram array. For example, from 1 second of audio signal the size of the spectrogram array is 94x100. When the spectrogram array is completed for each single word, Discrete Cosine Transform DCT is applied using equation (3.1):

\[ y(k) = \frac{1}{\sqrt{N}} u(k) \sum_{n=0}^{N-1} x(n) \cos \left( \frac{(2n+1)\pi k}{2N} \right) \quad \text{for } k = 0, 1, \ldots, N-1 \]

\[ u(k) = \begin{cases} 1 \sqrt{2}, & \text{if } k=0 \\ 1, & \text{otherwise.} \end{cases} \]

(3.1)

Only the first 30 values of DCT are used to create cepstral coefficients array and the size of the matrix will be (number of frames x 30). The outputs of DCT are real values with assumption of periodic and symmetrical values. Furthermore, the DCT basis sequences are Cosines (Theis et al., 2010). These Cosine Coefficients are provided by Twiddle built-in function. The dynamic allocation of memory in C programming (malloc) is used to solve the memory problems (no need to declare the size of the array).

3.3 Training and Testing stages.

An ADSP-BF561 EZ-KIT provides four push buttons for general purpose IO. These push buttons are labeled SW6 to SW9 which are connected to the programmable flags PF8-5. A FI00_FLAG_D register is used to read the status of each button. If the button is pressed-on, the corresponding bit of register reads 1 and when it is released, the bit reads 0. The SW4 DIP switch is used to establish the connection between the push buttons and the flags input (ANALOG DEVICE, 2008).
In this assignment, only the first (SW6) and the second (SW7) buttons are used to switch between the two stages. The first push button is programmed for training stage and the second is programmed for testing stage. After the input signal is processed through the steps as shown in figure 3.1 till DCT, the system will check which button was pressed. If the first button was pressed, the system will store the cepstral coefficients array in the allocated memory location depending on the order of the words. Each of the four templates is stored in specific section of the internal memory L2. However, if the second push button is pressed, the testing process will start and the cepstral coefficients of the live signal will store in specific location in the internal memory L2 and the DTW algorithm process will start.

3.4 Memory allocation.

An ADSP-BF561 contains two types of data memory which are internal and external memory. The internal memory consists of two sections L1 data memory and L2 (SRAM). In L1, there are two banks and each bank is divided into 4 sub-banks of 4KB. The total size of L1 section in each core 64KB (only one core is used). Moreover, L2 memory SRAM consists of 8 separate 16KB sub-banks providing 128KB. However, external memory L3 provides 64MB SDRAM and 8 MB flash (ASYNC) (Pycock et al., 2012).

The input samples of each frame are stored in L1 (1024 samples) because the system is designed to process the data frame by frame and then store the final results of each word in one location. All results from applying the FFT until spectrogram calculations are stored in L1 section to increase the speed of the process. The templates of the four words and a live signal features are stored in L2 which need large size of memory. Only L1 and L2 are used to store the results of all parts of the process. To reduce the size of memory which is needed for each step, fract16 number representation is used instead of floating-point format.

3.5 Samples of output results.

This section shows some of the output results during running process. All process steps were done on the word “one” and the results were as below:

![Figure 3.3: Sample of input_arr.](image-url)
Figure 3.4: Sample of fft_in.

Figure 3.5: Sample of fft_out.

Figure 3.6: Sample of fft_mag.

Figure 3.7: Sample of fft_log.
4. Direct Memory Access DMA.

An ADSP BF561 has two DMA controllers each one has 12 peripheral and 4 memory DMA channels (Pycock et al., 2012). DMA2 is used and the configuration was done in Initialize1.c file. Channel 4 is used to SPI transmit while channel0 is used to Sport receive and channel1 is used to Sport transmit.

5. Project files.

Most of the existing files in DSP_Lab1_exp5toStudDone are used in this assignment. Main codes are existed in data_process1.c file which includes the process steps from receiving audio signal till store it and applying DTW algorithm. Comments are used to show any modification or additional codes were done through preparation process to run the kit. To use the push buttons, two more files are used which are ezkitutilities.h and ezkitutilities.c and these files are existed in examples folder. The list of the project files is:

**Header files:** DSP project.h and ezkitutilities.h.

**Source files:** DSP project.c, ezkitutilities.c, Process_data1.c, ISR1.c and Initialize1.c.
6. **End point detection:**

6.1 **Method**

In order to calculate the features of isolated spoken words, it is very important to know that when the word is starting and when it is finishing. The method I used for this purpose is one so called “Energy based end point detection”. In this method, we assumed that after pressing the button to start our speech there is no voice signal up to 200ms. As our audio signal is sampled at 48 kHz frequency and each frame consists of 1024 samples (Pycock et al., 2012), so we assume that roughly for first 10 frames our audio signal is consists of noise and at least our word starts after that as shown below in figure 6.1.

![Graphical representation of an audio word](Meng, 2004).

Threshold energy is measured based on these first 10 frames to detect the actual starting and ending of the word. In our ADSP programme we calculated threshold energy two times greater than average background noise energy and so determined the isolated spoken word as follow:

- Calculated Energy $energy[i]$ of each sample in a frame by taking square of its amplitude level.
- Found average frame $f_{energy}$ by dividing the sample energy by the total no. of samples in a frame i.e. $current\ size=1024$.
- Measured total energy of first 10 frames and put it in a buffer $energy\_cal[]$.
- Found average energy $avg\_energy$ of the first 10 frames by dividing $energy\_cal[]$ by 10.
- Determined the threshold energy $thre\_energy$ by multiplying $avg\_energy$ by 2.
- A comparison is made between $f\_energy$ and $thre\_energy$ to decide that when our word is starting and when it is finishing.

In this assignment we stored templates for four isolated spoken words. The following flow chart of my DSP code shows the sequences of the procedure I used for this purpose.
As shown in the flow chart above, checks have been made to decide when to start and stop calculating features. When the word starts, a counter starts counting the number of words which is helpful later on in this programme in storing the templates.

### 6.2 Data Flow

Some of the settings for AD1836 codec have been taken from the existing DSP project “DSP_Lab1_exp5toStudDone” and new software for threshold energy calculation is written by myself. After ADC, the samples of data are received from pins of AD codec. In this part of our DSP code data is accessed and processed by using following for temporary storage of data in data transportation:

- **sRxBuffer1[]**: This buffer receives samples from the audio codec AD1836 registers iCodec1836TxRegs[] through slot0 defined as INTERNAL_ADC_L0 each time it is filled.
- **sCh0LeftIn[]**: A set of 32 samples are received by this buffer from sRxBuffer1[] each time it is filled up. A built in circular buffer __builtin_circindex is used to get the new required sample values from the input buffer.
- **input_arr[]**: 1024 samples are copied in this buffer from sCh0LeftIn[] every time when it filled. Again the same circular buffer is used to get the next 32 samples from the input buffer and a use of counter is made to start processing the frame once it has received 1024 samples successfully.
- **energy[]**: This buffer holds the squares of every single sample in a frame.
- **energy_cal**: The purpose of this buffer is to add up all of the values in energy[] and store the average of them.
- **thre_engy**: This buffer stores the threshold value of the signal energy and is called later on to make a decision that when our word is started and when it is finished.
6.3 Memory Requirement and Allocation

As we have a limited amount of memory available in ADSP-BF561 kit grouped in sections L1 (100Kbytes) being the smallest but faster in access, L2 (128Kbytes) to L3 (64Mbytes) being the slowest but big in amount (Pycock et al., 2012).

As we are using 16-bit word length which is 2 byte, the memory required by an array of N elements is 2*N. The buffer sCh0LeftIn[] only contains 32 elements at the time. That means it requires a memory of 64 bytes width. As this array is more frequently accessed in the programme we did not force it to be in a specific memory location. The buffers input_arr[] and energy[] reside 1024 samples in each so each of them need 2,048 Kbytes of memory. This is a significant amount of memory for L1 so L2 SRAM is allocated for them. The buffers like energy_cal and thre_engy are only one figure each so they only require a few bytes each. These variables are kept in L1 data bank a.

7. Dynamic time warping

7.1 Method

In this part of our ADSP programme, the function DTW_ISW takes four arguments array0, array1, fn0 and fn1 being the matrix of the cepstral coefficients of live word, the matrix of the cepstral coefficients of one of the stored templates, number of rows (frame numbers) in array0 and number of rows (frame numbers) in array1, respectively.

A distance matrix d[][] is calculated by summing the absolute difference between the respective elements in each row of the two matrixes. This gives the matrix d[][] dimensions as fn0 by fn1 (Pycock, 2012). Taking the given test matrixes as arguments my code yielded the following result which matches the correct result provided:

Figure 7.1: Out of C code for distance matrix

In the next stage, cumulative distance matrix D[][] is computed by adding the respective element from d[][] to the minimum one of the three elements back in neighbour. The resultant matrix has the same dimensions as d[][] (Pycock, 2012). This function returns the last element of the D[][] normalised with the summation of total number of rows and columns of the matrix i.e. D[fn0][fn1]/(fn0+fn1). This value gives the minimum distance between the two matrixes. The flow chart given below shows the sequence of the programme.

Figure 7.2: Flow chart of DTW.


8. TV Display

8.1 Method

We get four return values $x_1, x_2, x_3$ and $x_4$ by calling $dtw_isw$ four times comparing live word with each of the four stored templates. Some parts of an existing project “SRGP_bf561” were used to setup TV display and new programme is written to display the recognised word. After having four values $x_1, x_2, x_3$ and $x_4$ in our process data function, the function main2 is called from the attached file srgp.c. Rest of the code for display is written on this file. This was an existing file in the project mentioned above where I added new functions decision1, decision2, decision3 and decision4 and amended some of the existing ones like welcome, main2 and ShowTest and structure sTest as can be seen from the code files attached.

The functions decision1, decision2, decision3 and decision4 contain the four possible massages one, two, three and four to be displayed. These are the words we spoke in microphone and stored the templates for. The structure variable sTest TESTS[] is doing two jobs: first it contains the headings of the display messages “Isolated Spoken Word Recognition” and second the positions of the functions which provides the index value in this array that is 1 for decision1, 2 for decision2, 3 for decision3 and 4 for decision4. The function ShowTest(int) takes the index value from sTest TESTS[] as argument and actually prints the message on TV display. The function main2 sets up the coordinates of display for message, configures PPI port, enables data flow and decides which message is to be printed.

This function main2 is called in process_data function. In first step, it checks if $x_1$ is greater than $x_2$, $x_3$ and $x_4$, ShowTest[1] is called which prints the message from buffer sTest TESTS[] at index 1. This returns the message from decision1 that is “one”. Second condition else if $x_2$ is less than $x_3$ and $x_4$ is checked and if true, second message “two” is printed in a similar way as described above and so on third and fourth messages are printed. The following flow chart shows the code written for display purpose.

![Flow chart of C code for TV display.](image)

Figure 8.1: Flow chart of C code for TV display.
Some of the test display results are shown in the figures below.

**Figure 8.2: Display Test Results**

8.2 Video Codec Configuration:

Most of settings for video configurations have been taken from the pre-existing project mentioned above. Hardware connections are explained in section#1. For software configurations, functions settings are given in `srgrp.c` file attached. These functions set input buffers like `ADI_DEV_BUFFER`, process buffers like `buf[16]`, DMA display graphics, display coordinates, display colour, font size, text positions and so on. To explain all of these functions is impossible here in this short report, rather code files are provided for more details.

9. Performance Analysis and Experimental Results

In real time implementation of the programme, after running the programme and pressing push button SW6 (testing stage), we were speaking four isolated English words “one”, “two”, “three” and “four” in once. Right after saying “four”, we were pressing the second button SW7 (Pycock, 2012) and speaking any one of these four words (say we said three) again. The expected display word (“three”) could not achieve successfully but individual outputs of the parts described above were achieved successfully with small deviations from the correct values. Some of these results are shown in the figures below as short picture segments and details are given in the files attached.

**Figure 9.2: Programme Execution Results**
Figure 9.3: Number of Programme Execution cycles

As can be seen $d[]$ and $D[]$ are only one row, this is because they are declared as pointers and are convertible in matrix forms.

The output time delay of the programme is a function result precision, computational complexity, processing data size and processor power. We converted our floating pint values into \textit{fract16} type and integer values into \textit{short} to face minimum delay on the cost of precision in results. As all of the computations in programme parts described above are quite simple and they do not affect the performance of the programme. The only delay is in \textit{for loops} but not a big deal as their number of iteration is not very huge (e.g. mostly 30 and ~60 and once 1024). As mentioned in section 2.1.4, data size of the buffers used in these parts of programme (e.g. $d[]$ and $D[]$ are~ 7.2 Kbytes, \textit{cepstral[]} are 3.6 Kbytes, \textit{input[]} and \textit{energy[]} are 2.048 Kbytes) is quite reasonable as compared to the available memory and processing power of ADSP-BF561 (600 MHz clock rate ) (Pycock, 2012). Because of these reasons our code performance is alright and it executes on few msec. (927591121 number of cycles shown above in Hexadecimal). That is why optimization i.e. allocating various memory sections adding DMA channels and setting optimization settings built-in in visual DSP++ makes only a miner difference in time delay and performance.

10. Conclusion

EDSP programming and its implementation using ADSP BF-561 for Isolated spoken word recognition has been carried out in this report. Most of the targets are achieved successfully with some precision errors. Main challenging area we faced in this work was system software configurations. To deal with this challenge we consulted with the pre-existing projects as already mentioned in this reports and we have good understanding of the system. Efficiency in both sections i.e. front end analysis and DTW, is attained by using appropriate C code and improved by using some optimization technologies. The other challenge we dealt with was floating point computation and complex functions in front end analysis. As BF 561 is a fixed point supporting DSP so we had to pay a cost of result precision in optimization of our programme in front end calculations. However, in as mentioned in section 2, there were no complex function and any big issue regarding to floating point computations in end point detection, DTW and TV display. In future researches, more work is needed to carry out in this field for precision in results with minimum execution time.
References.


Online version available at:

http://www.knovel.com/web/portal/browse/display?_EXT_KNOVEL_DISPLAY_bookid=3250&VerticalID=0. [Accessed on 1 April 2012].