LPC10 Implementation on a Texas Instruments TMS320C6713 Digital Signal Processor

Submitted By:

Manish Bansal
Garima Sirvastava
Alexander Pasharikov

Project Abstract

The goal of this project is to implement a real-time speech coding algorithm. This project is particularly complex because it has both hardware and software components to it. The real-time nature of the project will be implemented by a Texas Instruments TMS320C6713 Digital Signal Processor (DSP). The DSP shall receive speech data, process it, and then immediately play it out through speakers. The speech coding algorithm that shall be implemented is inspired by the LPC10 Federal Standard 1015; however, it will not necessarily be implemented according to the exact specifications. Instead, it will be a simplified version of the LPC10 algorithm due to the limited time available to complete the project (approximately two weeks). This project will present a sufficient technical challenge due to the real-time implementation aspect. All of the algorithms will be programmed onto the DSP chip using the C programming language. Once the input speech data is coded, it will be decoded and then played out through a speaker. The objective is not to compress the speech data for transmission, but to achieve the best quality of speech and compare the quality to other speech coders. Speech quality will be judged based on intelligibility and naturalness.

Hardware and Software

The hardware used for this project was the TMS320C6713 DSP Starter Kit (DSK) board shown below. Sitting at the heart of the DSK is TMS320C6713 DSP. It is a 32 Bit Floating Point DSP with a clock rate range of 167 - 225 MHz. When configured to run at the highest clock rate, the DSP can deliver 1800 Million Instructions Per Second (MIPS). With the given specifications, the mentioned DSP proves to be extremely powerful for an audio/speech processing algorithm.

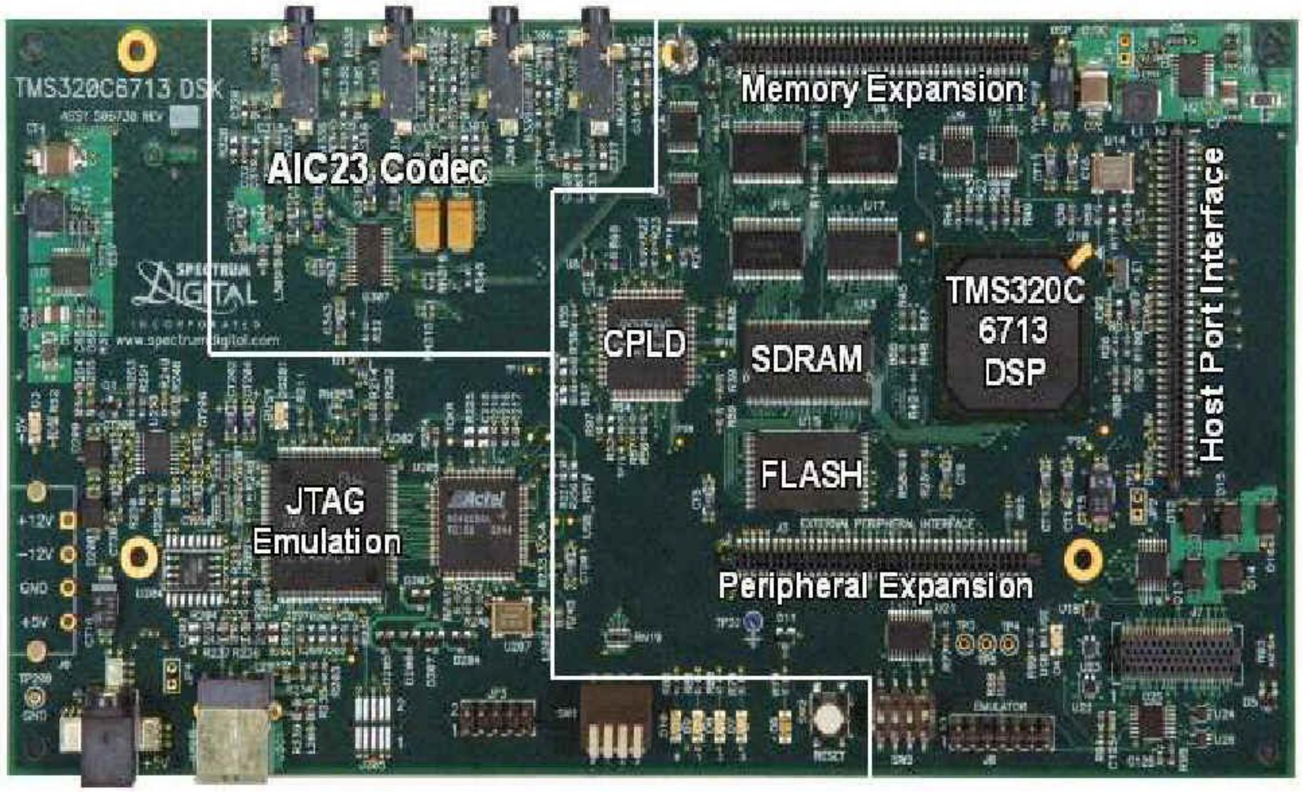

Fig.1. TMS320C6713 DSP Starter Kit (DSK)

A microphone was used to feed the audio data into the DSP via a 3.5 mm connector. The DSP board contains four 3.5 mm connectors that are interfaced by the AIC23 stereo codec. This codec is interfaced to the DSP using two serial ports McBSP0 and McBSP1. McBSP0 is configured in unidirectional mode to control the codec parameters by accessing its setting registers. On the other hand, McBSP1 is configured as a bidirectional port to transfer the audio data from and to the codec's ADC and DAC which are interfaced to input and ouput stereo ports. The DSP makes use of the Enhanced Direct Memory Access (EDMA) to transmit and receive the audio data through internal memory buffers which are accessible by the core. The synchronization between EDMA and the core is provided by implementing the buffers using Ping Pong Buffer technique. A block diagram describing the hardware implementation scheme is shown in Figure 2.

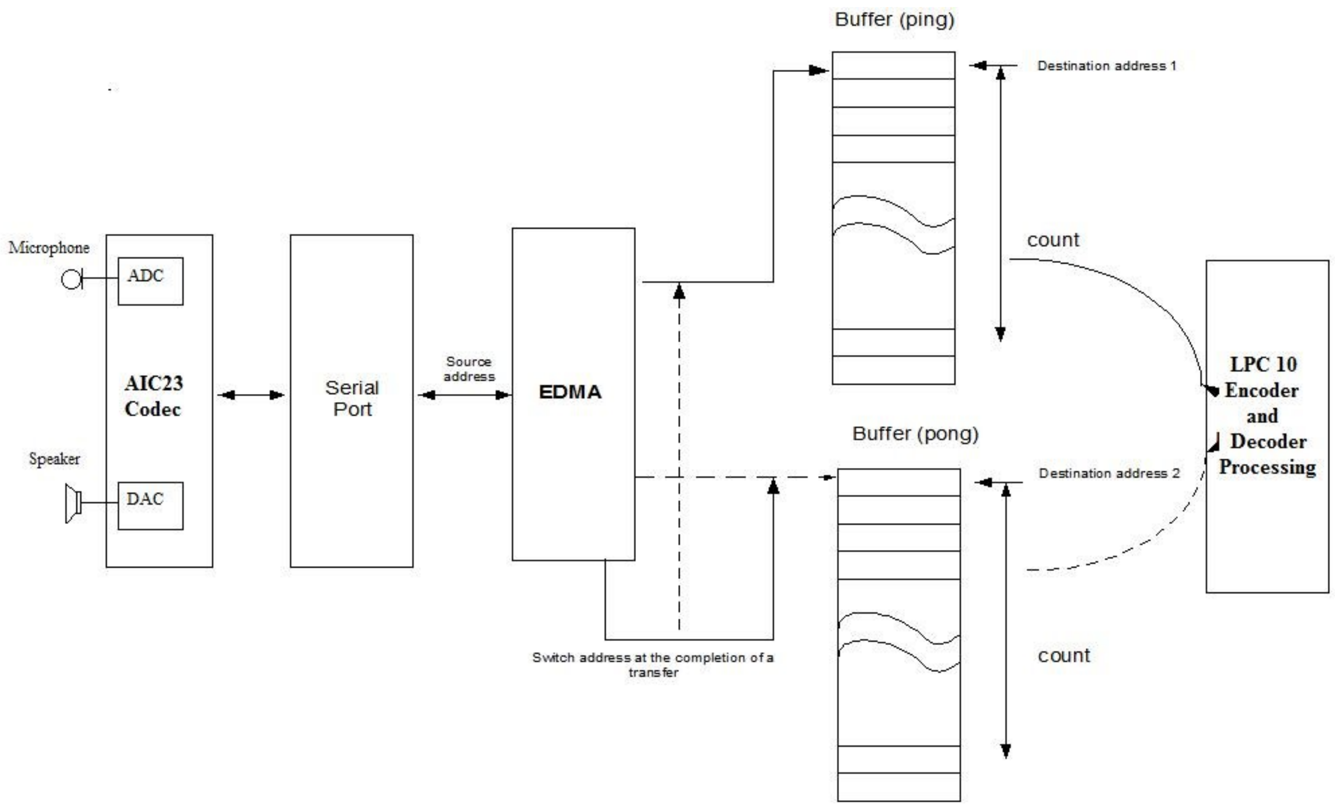

Fig.2. Implementation Block Diagram

Next, the DSP processes the data that is received in the Ping and Pong buffers in a alternating fashion. This processing involves encoding the data and then decoding it. The decoded data is then played out through the headphone jack of the AIC23. The software that was run on the DSP was developed using Texas Instruments' Integrated Development Environment Code Composer Studio (CCS). The Final CCS Project Folder is attached at the bottom of the website. The final version of the software that was run on the DSP is dsk_app.c. This project was originally an example that was included with the CCS software. It's function was to receive audio data, and play it out through the output port. It was beneficial for us to simply modify this software accordingly so that we could take advantage of the many functions and routines that were already written such as configuration of the DMA and McBSP. The DSP register settings were slightly modified and the all of our algorithms were inserted in the appropriate places. The algorithms that were implemented were all written by us in C, and they are described in the following sections.

The Speech Encoding Algorithm

The speech encoding algorithm is described in this section. The inspiration behind this algorithm is the LPC10 speech coder, which is defined by a United States Federal Standard 1015. The focus of this section is not to fully describe the LPC10 speech coder, but to describe the actual algorithm that was implemented on the DSP. Below is a summary of some of the main features of LPC10.

- 8 kHz sampling rate
- 10th order LPC predictor order
- 2.4 kbps data rate
- 22.5 ms frame length (180 sample frame length at 8 kHz)
- 54 bits per frame

This standard provided the speech compression at two levels. The first level of compression is achieved by processing the raw speech data frame to obtain 10 LPC coefficients, pitch, energy gain constant, and a voiced/unvoiced paramater. Comparing the data types of the coefficients and other parameters to actual speech data, this processing provided us a compression ratio of almost 1:8. This compression comes at the cost of losing the "naturalness" of the speech. A second level compression is achieved by quantization of these parameters. After quantization, the encoded frame consists of just 54 bits as tabulated below in Table 1.The 54 bits per frame are allocated in a specific way. Each number in the table below represents the number of bits used for a certain parameter. For the purposes of this project, it was unnecessary to quantize and pack the data in this compressed format because the intention is not to transmit the data.

Parameter	Voiced	Unvoiced
Pitch	6	6
Voiced/Unvoiced	1	1

Energy	5	5
LPC	41	20
Synchronization	1	1
Error Protection	---	21
Total	**54**	**54**

Table 1. Bit allocation in an LPC 10 Encoded Frame

The key difference between our implementation and original standard is that we do not quantize the parameters. This would provide better speech quality as compared to the quality achieved by the original standard, as we do not incur any quantization noise. As expected, this improvement comes at the price of longer encoded frame, which is 369 bits, as compared to 54 bits achieved by LPC10. In the end, though, speech quality will not be better than the LPC10 standard because of our simplified algorithms in which we neglect required parts of the specification. Similar to LPC10, our implementation also involves an 8 kHz sampling rate, 180 sample window size, and 10th order LPC analysis. A step wise approach to implement the encoder is summarized below.

1. The input speech data is windowed by a hamming window at 50% overlap.
2. Filter the speech data through a pre-emphasis filter.
3. The autocorrelation of the windowed data is calculated.
4. Using the autocorrelation, the LPC coefficients are found by the Levinson-Durbin Recursion algorithm. This method also produces an energy gain constant.
5. The speech frame is inverse filtered by the LPC coefficients to obtain the residue.
6. It is determined whether the windowed data is voiced or unvoiced. This process is done by analyzing two parameters. First, the energy of the speech data is compared to a threshold. Then, the autocorrelation of the residue is calculated and the ratio of the "after lag zero peak" value to the "lag zero peak" is compared to a threshold.
7. The autocorrelation of the residue is used to determine the pitch by detecting the After Lag Zero Peak detection.
8. The pitch, gain constant, voiced/unvoiced parameter, and LPC coefficients, combined, constitute a complete characterization of the speech frame. Thus, the windowed speech frame is completely encoded.

A block diagram of the above algorithm is shown in Fig. 3. All that is left to do is decode the data.

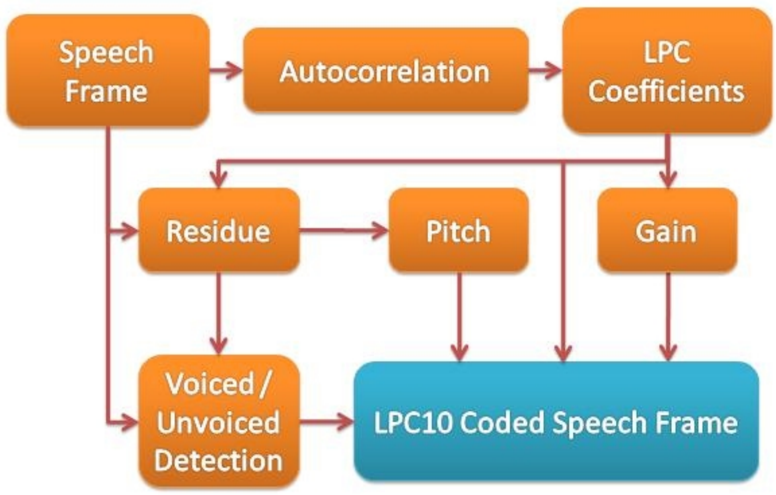

Fig. 3. Speech Encoder Block Diagram

The Speech Decoding Algorithm

Once the speech has been fully encoded, the necessary algorithms can be implemented to reconstruct the speech. The end result will be a synthetic speech that can be transmitted with as little bandwidth as possible if necessary. The hope is that the synthetically reproduced speech is an intelligible approximation to the original speech. The following steps explain the logical flow of the decoding algorithm.

1. Create the glottal excitation model - The LPC coefficients that characterize the frame will act as a vocal tract filter. That which is input to the filter is the excitation.
2. For voiced speech, the excitation will be a periodic signal of all zeros with impulses every pitch period. This is a simplified model of the glottal excitation. For unvoiced speech, the excitation will simply be white noise.
3. Inverse filter the excitation through the LPC coefficient filter.
4. The output of the vocal tract filter is further filtered through a de-emphasis filter.
5. The output of the de-emphasis filter is appropriately scaled as per the Energy Gain of the frame.

The block diagram for the speech decoder is shown below.

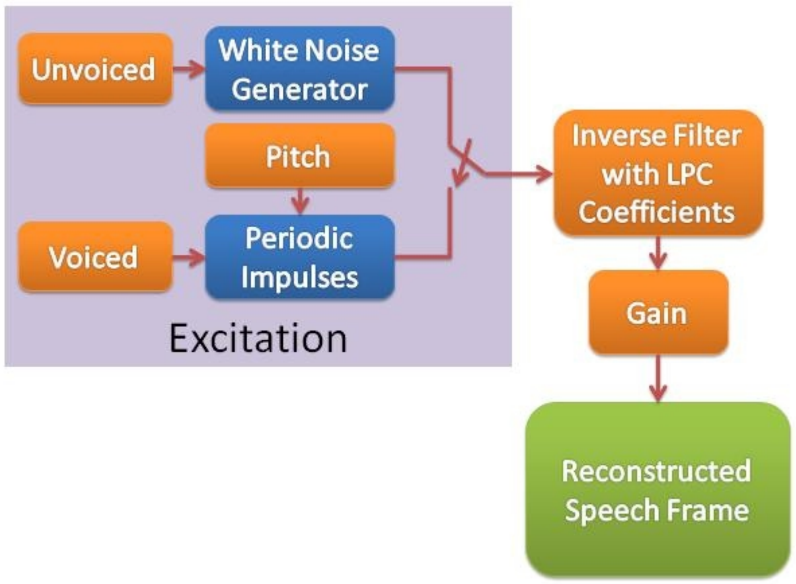

Fig. 4 Speech Decoder Block Diagram

MATLAB Simulation Results

The algorithms described above were first simulated using MATLAB. The MATLAB codes that were used are process.m and PowerMeasure.m. The results are shown below.

Original	male1.wav
Rectangular window with no overlap	Result1.wav
Hamming window with 50% overlap	Result2.wav
Hamming window with 50% overlap and a pre-emphasis filter	Result3.wav

Table 2. MATLAB Simulation Results

The pre-emphasis filter implemented above has a system function of (1 - 0.9375z^-1). These results were unnatural sounding, but they were still intelligible. There is some noticeable improvement with each new iteration, but the results still do not sound as good as an actual LPC10 coder. These results can be compared to the results from Arizona State University's speech coding demos. Next, the algorithms were implemented on the DSP.

Real-Time DSP Implementation Results

The following results were recorded from the output port of the DSP board. The quality of the recording equipment and recording environment was not ideal, but it can be heard that the algorithms were successfully implemented. The results are shown below.

	Adult Male	Adult Female
Original	alexnat.wav	garimanat.wav
Output of DSP	alexart.wav	garima_art.wav

Table 3. DSP Implementation Results

The actual algorithms and processes that were implemented on the DSP were not exactly equivalent to the MATLAB algorithms. The MATLAB simulations were performed so that there would be an approximate measure to compare our DSP results with. The actual algorithms that were implemented on the DSP were slightly more sophisticated and fine tuned for the specific function. Thus, the real-world implementation sounded better than the simulation results.

Conclusion

In conclusion, the end results were very satisfactory given the limited amount of time to complete the project. The overwhelming majority of the time was spent on developing the C code that ran on the DSP. Simple tasks proved to be difficult and complicated. This was primarily due to our limited experience with the C language and complex DSP architecture. In the end, we learned very much. Although the quality of speech was not as good as other speech coders, it was still surprisingly good considering that the output speech was artificially generated. It was expected, though, that the quality would not be as good as other coders. The main drawback in speech quality is when the speaker is a female or child because of the higher pitch voicing. The speech for an adult male sounded slightly distorted and unnatural, whereas the speech for an adult female sounded heavily distorted and artificial. The fact that female speech has a higher pitch is not even discernible when listening to the sound samples above. This may be evidence of the irreversible loss of quality associated with LPC analysis. Even the best LPC10 speech coders have an unnatural, robotic sound to the reproduced speech.

The LPC10 speech coder is a simple and efficient way to compress speech. The data rate is lower than any other speech coder that exists today, but this is at the cost of significant speech quality losses. We were very satisfied with our accomplishments. Given more time, the algorithms could have been tweaked to optimize the speech quality. Also, more sophisticated functions could have been implemented to improve the results. In the end, this project proved that a very low bandwidth speech coder can be implemented using relatively simple algorithms. Although the implementation on the DSP is rather difficult and time consuming, the experience that we gained was invaluable. This project really revealed the true power of a DSP and it's relevancy in real-world applications.

References

1. Andreas S. Spanias, "Speech Coding: A Tutorial Review", Proceedings of IEEE (1994).
2. Professor Joseph Picone, "Lecture 16 -ECE 8463: Fundamentals of Speech Recognition", Department of Electrical and Computer Engineering, Mississippi State University.
3. Documentation on TMS320C6713 DSK.
4. Statistical And Model Based Approach To Unvoiced Speech Detection, Krithika Giridharan, Brett Y. Smolenski and Robert E. Yantorno
   Temple University/ECE Dept.
5. Soo Hyun Bae, Presentation on "LPC-10 2.4 kbps Federal Standard in Speech Coding", 2004

FinalSpeechProject.zip (397k)	alexander pasharikov, Apr 22, 2009 12:30 AM	v.1
PowerMeasure.m (0k)	alexander pasharikov, Apr 19, 2009 8:57 PM	v.1
Result1.wav (90k)	alexander pasharikov, Apr 18, 2009 2:09 PM	v.1
Result2.wav (89k)	alexander pasharikov, Apr 18, 2009 2:28 PM	v.1
Result3.wav (89k)	alexander pasharikov, Apr 19, 2009 9:11 PM	v.1
SpeechProjectPresentation.pptx (323k)	alexander pasharikov, Apr 24, 2009 12:47 PM	v.1
alexart.wav (831k)	alexander pasharikov, Apr 19, 2009 9:48 PM	v.1
alexnat.wav (431k)	alexander pasharikov, Apr 19, 2009 9:48 PM	v.1
dsk_app.c (29k)	alexander pasharikov, Apr 22, 2009 12:30 AM	v.1
garima_art.wav (677k)	alexander pasharikov, Apr 19, 2009 9:49 PM	v.1
garimanat.wav (121k)	alexander pasharikov, Apr 19, 2009 9:49 PM	v.1
male1.wav (90k)	alexander pasharikov, Apr 17, 2009 5:18 PM	v.3
process.m (2k)	alexander pasharikov, Apr 19, 2009 8:56 PM	v.1

Comments

You have no permission to add comments.