*2012 International Conference on Information and Network Technology (ICINT 2012)*
***IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore***

# Design and Implementation of ADPCM Based Audio Compression Using VHDL

Dhanashri Gawali [+] , Nirija Varma, Tejashri Dixit and Sneha Mandowara

E&TC Department, Maharashtra Academy of Engineering, Alandi(D), Pune

**Abstract.** For Internet-based voice transmission, digital telephony, intercoms, telephone answering machines, and mass storage, we need to compress audio signals. ADPCM is one of the techniques to reduce the bandwidth in voice communication. More frequently, the smaller file sizes of compressed but lossy formats are used to store and transfer audio. Their small file sizes allow faster Internet transmission, as well as lower consumption of space on memory media. However, lossy formats trade off smaller file size against loss of audio quality, as all such compression algorithms compromise available signal detail. This paper discusses the implementation of ADPCM algorithm for audio compression of .wav file. It yields a compressed file ¼ of the size of the original file. The sound quality of the audio file is maintained reasonably after compression.

**Keywords:** ADPCM, audio compression, VHDL.

## 1. Introduction

Different compression techniques for still pictures include horizontal repeated pixel compression (pcx format), data conversion (gif format), and fractal path repeated pixels. For motion video, compression is relatively easy because large portions of the screen don't change between each frame; therefore, only the changes between images need to be stored. Text compression is extremely simple compared to video and audio. One method counts the probability of each character and then reassigns smaller bit values to the most common characters and larger bit values to the least common characters. However, digital samples of audio data have proven to be very difficult to compress; as these techniques do not work well at all for audio data. The data changes often and no values are common enough to save sufficient space. The sampling frequencies in use today are in range from 8 kHz for basic speech to 48 kHz for commercial DAT machines. The number of quantizer levels is typically a power of 2 to make full use of a fixed number of bits per audio sample. The typical range of bits per sample is between 8 and 16 bits. The data rates associated with uncompressed digital audio are substantial. For audio data on a CD, for example, which is sampled at 44.1 kHz with 16 bits per channel for two channels, about 1.4 megabits per second are processed. A clear need exists for some form of compression to enable the more efficient storage and transmission of digital audio data. Although lossless audio compression is not likely to become a dominating technology, it may become a useful complement to lossy compression algorithms in some applications. This is because, lossless compression algorithms rarely obtain a compression ratio larger than 3:1, while lossy compression algorithms allow compression ratios to range up to 12:1 and higher[1]. For lossy algorithms, as compression ratio increases, final audio quality lowers. For digital music distribution over the Internet, some consumers will want to acquire the best possible quality of an audio recording for their high-fidelity stereo system. However lossy audio compression technologies may not be acceptable for this application.

Voc File Compressions technique simply removes any silence from the entire sample. This method analyzes the whole sample and then codes the silence into the sample using byte codes. Logarithmic

[+] Corresponding author. Tel.: +91-20-30253500.
*E-mail address*: dhanashree.gawali@gmail.com.

compression such as μ-law and A-law compression only loses information which the ear would not hear anyway, and gives good quality results for both speech and music. Although the compression ratio is not very high it requires very little processing power to achieve it. This method is fast and compresses data into half the size of the original sample. A Linear Predictive Coding (LPC) encoder compares speech to an analytical model of the vocal tract, then throws away the speech and stores the parameters of the best-fit model. The MPEG compression is lossy, but nonetheless can achieve transparent, perceptually lossless compression. It may be observed that differential encoding schemes take advantage of sources that show a high degree of correlation from sample to sample. These schemes predict each sample based on past source outputs and only encode and transmit the differences between the prediction and the sample value. When a source output does not change greatly from sample to sample this means the dynamic range of the differences is smaller than that of the sample output itself. This allows the quantization step size to be smaller for a desired noise level, or quantization noise to be reduced for a given step size.

## 2. ADPCM Algorithm

The DVI Audio compression using ADPCM (ADPCM) algorithm was first described in an IMA recommendation on audio formats and conversion practices. ADPCM is a transformation that encodes 16-bit audio as 4 bits (a 4:1 compression ratio). In order to achieve this level of compression, the algorithm maintains an adaptive value predictor, which uses the distance between previous samples to store the most likely value of the next sample. The difference between samples is quantized down to a new sample using an adaptive step-size. The algorithm in [11] suggests using a table to adapt this step-size to the analyzed data. ADPCM has become widely used and adapted, and a variant of the algorithm performs voice encoding on cellular phones (allowing minimal data to be sent across the wireless network and increasing throughput).

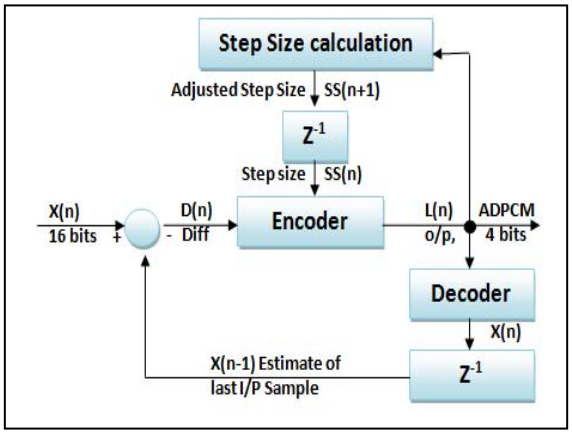


Fig. 1: (a) ADPCM encoding process Algorithm

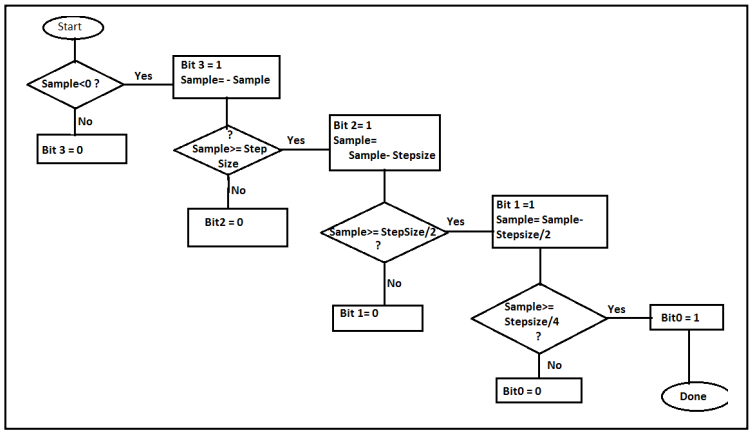


Fig. 1: (b) ADPCM encoding process Flowchart

Figure 1(a) shows a block diagram of the ADPCM encoding process. A linear input sample X(n) is compared to the previous estimate of that input X(n-l). The difference, d(n), along with the present step size, ss(n), is presented to the encoder logic. This logic, described below, produces an ADPCM output sample. This output sample is also used to update the step size calculation ss(n+l), and is presented to the decoder to compute the linear estimate of the input sample. The encoder accepts the differential value, d(n), from the comparator and the step size, and calculates a 4-bit ADPCM code. The following is a representation of this calculation in pseudo code:

Let B3 = B2 = B1 = B0 = 0

If (d(n) < 0) then B3 = 1 and d(n) = ABS(d(n))

If (d(n) >= ss(n)) then B2 = 1 and d(n) = d(n) -ss(n)

If (d(n) >= ss(n) / 2) then B1 = 1 and d(n) = d(n) -ss(n) / 2

If (d(n) >= ss(n) / 4) then B0 = 1

L(n) = (10002 * B3) + (1002 * B2) + (102 * B1) + B0

For both the encoding and decoding process, the ADPCM algorithm adjusts the quantizer step size based on the most recent ADPCM value. The step size for the next sample, n+l, is calculated with the equation, ss(n+1) = ss(n) * 1.1M(L(n))

This equation can be implemented efficiently using two lookup tables [11]. First table uses the magnitude of the ADPCM code as an index to look up an adjustment factor. The adjustment factor is used to move an index pointer located in second table. The index pointer then points to the new step size. This method of adapting the scale factor with changes in the waveform is optimized for voice signals. When the ADPCM algorithm is reset, the step size ss(n) is set to the minimum value (16) and the estimated waveform value X is set to zero (half scale). Playback of 48 samples (24 bytes) of plus and minus zero (10002 and 00002) will reset the algorithm. It is necessary to alternate positive and negative zero values because the encoding formula always adds 1/8 of the quantization size. If all values are positive or negative, a DC component would be added that would create a false reference level.

## 3. System Implementation

A number of different compression algorithms have been developed, and implemented on various platforms such as embedded symmetric multiprocessor data signal processing[3, 5, 6] platform, DSP processor[4], VLSI [2] etc. This paper includes design and implementation of audio codec using ADPCM technique. The platform used here for implementation is reconfigurable, based on FPGA, using VHDL.

### 3.1. System design

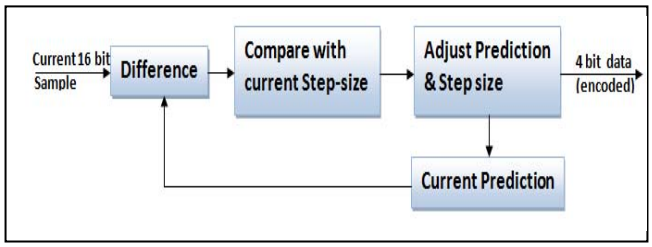


Fig. 2: (a). ADPCM encoder.

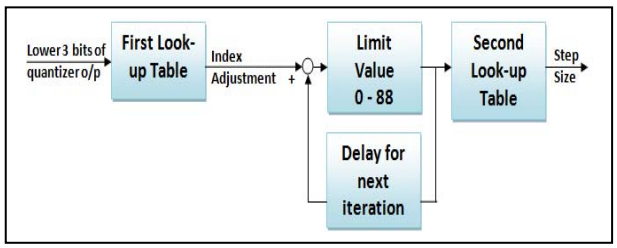


Fig. 2: (b). Block diagram of the step-size adaptation process

For simplicity, the figure omits details such as bit-stream formatting, the possible use of side information, and the adaptation block. Instead of representing each audio sample, an ADPCM encoder computes the difference between each audio sample and its predicted value. Note that the ADPCM encoder uses most of the components of the ADPCM decoder to compute the predicted values. The quantizer adapts the step size based on the current step size and the quantizer output of the immediately previous input. The three bits representing the number of quantizer levels serve as an index into the first lookup table whose output is an index adjustment for the second lookup table. This adjustment is added to a stored index value, and the range-limited result is used as the index to the second lookup table. The summed index value is stored for use in the next iteration of the step-size adaptation. The output of the second lookup table is the new quantizer step size.

### 3.2. System implementation using VHDL

The system here refers to the integrated top level module. The top-level module is basically interconnection of ADPCM Encoder, Audio memory and Step-table memory.

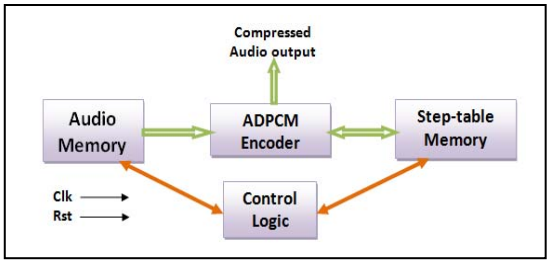


Fig. 3: Block diagram of ADPCM encoder

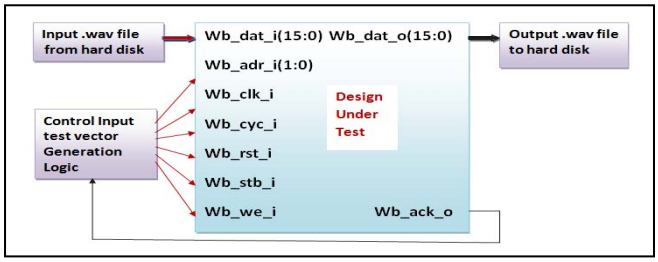


Fig. 4: Test environment for system

The top module controls the read-write operation of audio samples be encoded, updating of step-table memory and operations of ADPCM encoder. The output of the system is also of same size as that of input audio sample but a stream of compressed audio samples is given by encoder module. The system converts each 16 bit audio sample into 4 bit compressed audio format. The encoder module executes ADPCM algorithm. It has been implemented as a state machine. Input to the module is 16 bit audio sample. The values of step table are taken from Step-table memory. The output of this module comprises of predicted value of next audio sample, final step index value and 4-bit audio sample which is the compressed version of 16-bit audio input. The Audio memory module is used for storage of audio file samples which are in .wav format. Step-table memory module is used to store the index values which are used in execution of compression algorithm.

# 4. System Testing

The design description is done with VHDL. Design Simulation was carried out to check timing and functionality of design at various stages. As shown in figure 4 a test environment is created with VHDL test bench. It consists of testing logic, input and output buffer which is responsible for taking a .wav file from the hard disk as a input to the system and buffering out the compressed file from the system respectively. The design has been further synthesized and implemented in Xilinx FPGA, Spartan II, Device 2S200-5pq208.

# 5. System Results

## 5.1. Simulation result

The simulation of the program is carried out and the result is shown in Figure 5(a). It represents the system operation for the corresponding input test conditions.

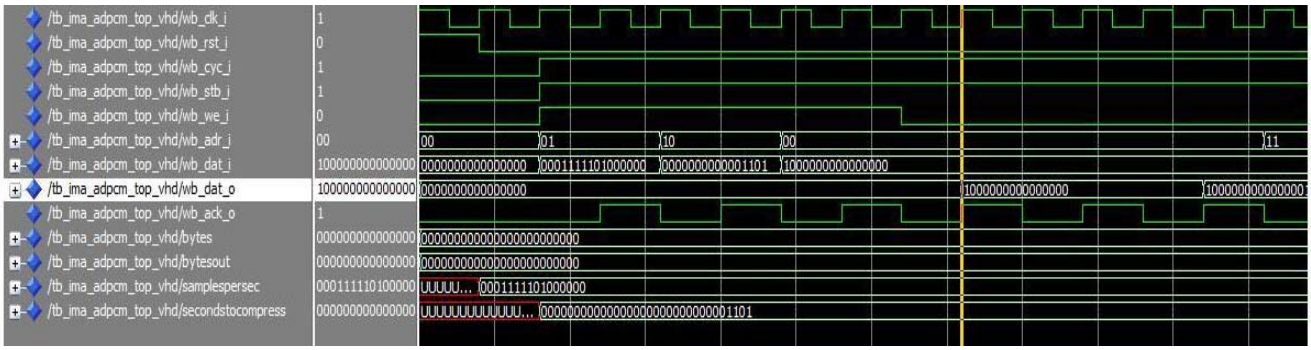


Fig. 5(a) : System functionality snapshot

System implemented is able perform two functions, compress an audio file in .wav format to ¼ of its original size and maintain reasonable sound quality of the audio file even after compression.

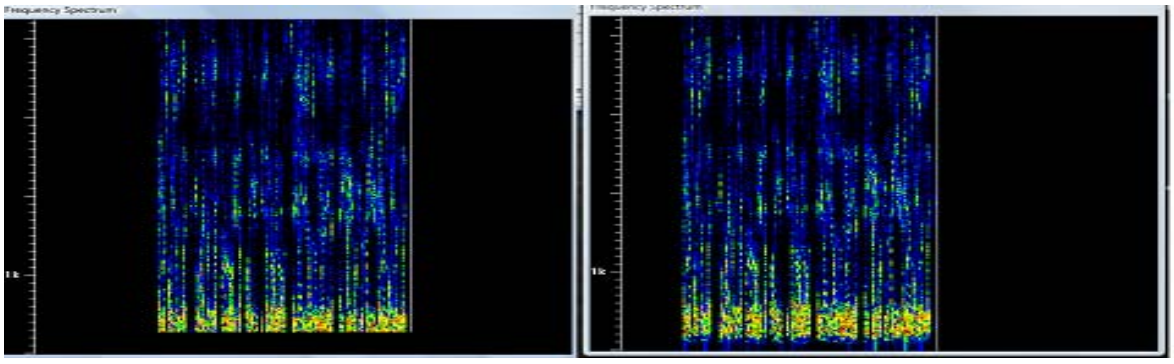

Fig. 5(b) : FFT of input and output files.

Table I shows the differences in properties for input and output audio files i.e. audio file before and after compression. The bit rate and the audio sample size have been reduced to ¼ with no difference in audio sample rate. For validating the sound quality of compressed file, FFT(fast furrier transformation) of both input and out file have been compared using frequency analyzer v.20 software From the FFT(figure 5.b) it may be predicted that the quality of the sound is almost same before and after compression.

Table. 1: Audio properties

| S.N. | Property | Input Audio | Output |
|---|---|---|---|
| 1 | Audio format | PCM | IMA ADPCM |
| 2 | Channels | 1 (mono) | 1 (mono) |
| 3 | Audio | 8 KHz | 8 KHz |
| 4 | Audio | 16 bit | 4 bit |
| 5 | Bit Rate | 128 kbps | 32 kbps |

Table. 2: Device Utilization of design

| S.N. | Resources | Used | Available |
|---|---|---|---|
| 1 | Number of Slices | 748 | 2352 |
| 2 | Number of Slice Flip | 369 | 4704 |
| 3 | Number of 4 input | 1409 | 4704 |
| 4 | Number of bonded | 40 | 144 |
| 5 | Number of BRAM | 2 | 14 |
| 6 | Number of GCLKs | 1 | 4 |

### 5.2. Synthesis result

Device Utilization for Spartan II, 2s200pq208 FPGA is shown in Table II. Approximately 25% device has been utilized by the system. The evaluated maximum operating clock frequency was 44.203 MHz.

## 6. Conclusion

Implementation of ADPCM algorithm for audio compression of .wav file using VHDL resulted in a compressed file, ¼ of the size of the original file. The sound quality of the audio file is maintained reasonably after compression. The compression leads to better storage scheme which can further be utilized for many applications in communications. The system can be used for real time audio streaming in future streaming media, such as audio or video files sent via the Internet. Audio/video data can be transmitted from the server more rapidly with this system as interruptions in playback as well as temporary modem delays are avoided. The main application intended for such system is to real time streaming however it could be used in various other application such as Voice Storage, wireless communication equipments etc.

## 7. References


[1] Mat Hans and Ronald W. Schafer “Lossless compression of digital audio” IEEE SIGNAL PROCESSING MAGAZINE, 1053-5888/01, JULY 2001, pp 21-32

[2] Peilin Liu,Lingzhi Liu, Ning Deng, Xuan Fu, Jiayan Liu, Qianru Liu,” VLSI Implementation for Portable Application Oriented MPEG-4 Audio Codec” IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007. Pp- 777 - 780

[3] Bochow, B. ; Czyrnik, B. ; “Multiprocessor implementation of an ATC audio codec” International Conference on Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989, pp. 1981 - 1984 vol.3

[4] Jing Chen ; Heng-Ming Tai ; “Real-time implementation of the MPEG-2 audio codec on a DSP” IEEE Transactions on Consumer Electronics, Volume 44 , Issue 3 , pp. 866 – 871

[5] Gurkhe, V. ; “Optimization of an MP3 decoder on the ARM processor” Conference on Convergent Technologies for Asia-Pacific Region TENCON 2003, pp. 1475 - 1478 Vol.4

[6] Kyungjin Byun ; Young-Su Kwon ; Bon-Tae Koo ; Nak-Woong Eum ; Koang-Hui Jeong ; Jae-Eul Koo ; “Implmentation of digital audio effect SoC ” IEEE International Conference on Multimedia and Expo, 2009. ICME 2009, pp. 1194 – 1197

[7] Ouyang Kun ; Ouyang Qing ; Li Zhitang ; Zhou Zhengda ;” Optimization and Implementation of Speech Codec Based on Symmetric Multi-processor Platform” International Conference on Multimedia Information Networking and Security, 2009. MINES '09. Pp. 234 – 237

[8] K. Brandenburg and G. Stoll, "The ISO/MPEG-Audio Codec: A Generic Standard for Coding of High Quality Digital Audio," Preprint 3336, 92nd Audio Engineering Society Convention, Vienna1992.

[9] Rabiner and R. Schafer, Digital Processing of Speech Signals (Englewood Cliffs, NJ: Prentice-Hall, 1978).

[10] M. Nishiguchi, K. Akagiri, and T. Suzuki, "A New Audio Bit Rate Reduction System for the CD-I Format," Preprint 2375, 81st Audio Engineering Society Convention, Los Angeles 1986.

[11] IMA Digital Audio Focus and Technical Working Groups, “Recommended Practices for Enhancing Digital Audio Compatibility in Multimedia System: Revision 3.00,” IMA Compatibility Proceedings, Vol. 2, October 1992.