A LOW COMPUTATIONAL PREDICTOR COEFFICIENT ALGORITHM FOR ADPCM IMPLEMENTATION OF PORTABLE RECORDING DEVICES

C. Boonyakitmaitree K. Nandhasri and J. Ngarmnil
Department of Electronic Engineering,
Mahanakorn University of Technology, 10530, Thailand
Email: jitkasam@mut.ac.th

Abstract - This paper proposes a modification of ADPCM algorithm to be suitable for hardware implementation of portable audio devices. This is accomplished by a simplification of predictor coefficient algorithm which normally requires a high complexity correlation algorithm, while the sound quality measured as SNR remains relatively the same to those of the original ADPCM. The less complicated calculation results in a reduction of the chip area and power consumption, thus it is suitable for pocket solid-state speech and audio recorder. Simulations on MATLAB verify the performances and experimental results on XILINX FPGA synthesis confirming the functionalities and significant reduction of device count by 63% of the implementation by the original algorithm.

I. INTRODUCTION

On the emergence of portable and mobile computing, low-power circuits are significantly demanded by applications of portable devices. MP3 players have been growing in popularity as a portable solid-state audio device due to the reduction of flash memory costs. However, these MP3 devices possess decoding-only or play-back only. There is hardly a portable device supporting MP2/MP3 encoding function because the MP3 encoding algorithm is very complex and demands a huge computational capability which is equivalent to the use of several DSP processors. The main task leading to the high complexity of MPEG encoder is from that the Psycho-Acoustic Model (PAM) and the iterative bit allocation consist of numerous hardware consuming computational functions such as logarithmic, exponential and power. These operations are not possible to be implemented as an ASIC or DSP device. However, there is still demand of portable devices with recording capability of high quality sound and music. This paper presents a simplification of adaptive predictor and coefficient algorithm to be suitable for an implementation of ADPCM algorithm [1]-[5], which offers high compression ratio, high signal quality but with less hardware consuming. The technique is suitable for implementation of a low power sound recording device for consumer portable applications such as voice memo and voice recorder on mobile phone and PDA etc.

II. BASIC CODING ALGORITHM

Pulse code modulation (PCM) samples an input signal using a fixed quantization to produce a digital representation. This technique, although simple to be implemented, does not take advantage of any redundancies in speech signals. There are no any linkage between the present input and the next samples. The value of the present input sample does not have an effect to the coding of future samples. Adaptive Differential PCM (ADPCM), on the other hand, uses an adaptive predictor, that adjusts according to the value of each input sample, and then reduces the number of bits required to represent the sampled data by at least factor of two. Another mean to sample speech is to use a model of the way people generate speech. In the Linear Predictive Coding (LPC) algorithm, the human vocal tract is modeled. Humans have an excitation source at the source of vocal tract and muscles along the tube is constricted which, in effect, shapes the waveform. People change the constriction points to make the various sounds. LPC uses a series of filters that accomplish a similar function. Sound reproduction can be very good and its performance is primarily limited by how well the excitation waveform can be reproduced. In the LPC algorithm, the filter coefficients and the excitation type are all that is needed to be transmitted which can be significantly less than the amount of information need to be transmitted for PCM methods. The reduced bandwidth requirements of LPC come at the expense of the large amount of processing power necessary for the algorithm. LPC works well for sending human speech sounds but not very well for music. The summary of algorithm comparison is shown in Table 1.

Table 1 Algorithm comparison

<table>
<thead>
<tr>
<th></th>
<th>PCM</th>
<th>DM</th>
<th>ADPCM</th>
<th>DPCM</th>
<th>ADPCM</th>
<th>LPC</th>
<th>MP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech quality</td>
<td>Very high</td>
<td>Low</td>
<td>Low</td>
<td>Adequate</td>
<td>Excellent</td>
<td>low</td>
<td>Excellent</td>
</tr>
<tr>
<td>Audio quality</td>
<td>Very high</td>
<td>Low</td>
<td>Low</td>
<td>Average</td>
<td>High</td>
<td>Very low</td>
<td>Excellent</td>
</tr>
<tr>
<td>Compression ratio</td>
<td>1:1</td>
<td>1:4</td>
<td>1:16</td>
<td>1:64</td>
<td>1:128</td>
<td>1:512</td>
<td>1:11</td>
</tr>
<tr>
<td>Complexity</td>
<td>Very low</td>
<td>Very low</td>
<td>Very low</td>
<td>Average</td>
<td>High</td>
<td>Very high</td>
<td>Ultra high</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Very low</td>
<td>Very low</td>
<td>Very low</td>
<td>Low</td>
<td>Average</td>
<td>Very high</td>
<td>Ultra high</td>
</tr>
<tr>
<td>Hardware consuming</td>
<td>Low</td>
<td>Low</td>
<td>Average</td>
<td>High</td>
<td>High</td>
<td>Very high</td>
<td>Ultra high</td>
</tr>
</tbody>
</table>

A choice of the compression algorithms to be used to implement an efficient portable recording device should mainly target on the best sound quality in relating to the data compression ratios and hardware complexity. Based on Table 1, we found that ADPCM and MP3 give comparable sound qualities with high data compression ratios in relative to the PCM and the rest algorithms. Complexity, directly reflecting to the hardware size and power consumption, is considered to determine a possibility that the algorithms could be implemented as integrated circuits. With the comparative characteristics, it is obviously known that the MP3 requires a much higher complexity than that of ADPCM. ADPCM block diagrams shown in Fig.1 are then an algorithm suitable for implementations.
where the coefficient $a_1$ and $a_2$ are shown in (2).

\[
\begin{bmatrix}
a_1 \\
a_2 \\
\end{bmatrix} = \begin{bmatrix}
 r_{ss}(0) & r_{ss}(-1) \\
 r_{ss}(1) & r_{ss}(0) \\
\end{bmatrix}^{-1} \begin{bmatrix}
r_{ss}(1) \\
r_{ss}(2) \\
\end{bmatrix}
\]

which is then rearranged to (3) and (4)

\[
a_1 = \frac{r_{ss}(0)r_{ss}(1) - r_{ss}(1)r_{ss}(0)}{r_{ss}^2(0) - r_{ss}(-1)r_{ss}(1)}
\]

\[
a_2 = \frac{r_{ss}(0)r_{ss}(2) - r_{ss}(-1)r_{ss}(1)}{r_{ss}^2(0) - r_{ss}(-1)r_{ss}(1)}
\]

and the autocorrelation $r_{ss}(m)$ is detailed in (5).

\[
r_{ss}(m) = \frac{1}{N} \sum_{n=1}^N s(n)s(n+m)
\]

where $m$ is a block size exhibiting the total number of samples in a block. $N$ is 1024 in this experiment. The autocorrelation sequence in (5) represents a correlation between present sample and future signal from $m$ consecutive samples. In this application, $m$ is denoted as $\{-1, 0, 1, 2\}$ as a result of a second order digital filter. It is noted in (3)-(5) that the equations require complex algorithms which are not suitable for implementations. Simplifications will be presented.

3.2 Simplification of Autocorrelation

To illustrate the simplification concept of the autocorrelation, we present an example in Table 2 assuming a block of input signal comprising seven samples denoting $n$ equal to $(0, 1, 2, \ldots, 6)$. The result of $r_{ss}(m)$ for $m=1$ and $m=-1$ are calculated in Table 3 for the next sample ($m=1$) and former sample ($m=-1$).

### Table 2 Sound example

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(n)</td>
<td>7</td>
<td>15</td>
<td>31</td>
<td>43</td>
<td>62</td>
<td>77</td>
<td>88</td>
<td>112</td>
</tr>
</tbody>
</table>

### Table 3 Autocorrelation sequence

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_{ss}(m)</td>
<td>465</td>
<td>1333</td>
<td>2666</td>
<td>4774</td>
<td>6776</td>
<td>9856</td>
<td>14560</td>
<td>57737143</td>
</tr>
</tbody>
</table>

The simplification starts from our notice on Table 3 that if we neglect the data '105' and '14560' by assume zero to these data, we can make the result of $r_{ss}(1)$ equal to those of $r_{ss}(-1)$ as shown in Table 5. Then our simplification is to neglect and assume zero to the first and the last data in the shade area of Table 2 which is then rewritten to Table 4. Then we can compute the autocorrelation sequence again as in Table 5, which is now the modified $r'_{ss}(m)$.

### Table 4 Modified sample

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(n)</td>
<td>0</td>
<td>15</td>
<td>31</td>
<td>43</td>
<td>62</td>
<td>77</td>
<td>88</td>
<td>112</td>
</tr>
</tbody>
</table>

### Table 5 Modified autocorrelation sequence

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r'_{ss}(m)</td>
<td>465</td>
<td>1333</td>
<td>2666</td>
<td>4774</td>
<td>6776</td>
<td>9856</td>
<td>0</td>
<td>25870</td>
</tr>
</tbody>
</table>

Authorized licensed use limited to: California State University Sacramento. Downloaded on March 03,2010 at 18:24:40 EST from IEEE Xplore. Restrictions apply.
From Table 5 it is clearly seen that if we neglect the first and the last points, the autocorrelation sequence result becomes $r_{ss}^*(m) = r_{ss}^*(-m)$. We can derive (5) to (6) which is then derived to (7). In general, the block size is normally large enough, for example with 20ms audio block sampling at 44.1 kHz, the size of the block will be 881 points. So (6) is simplified to (7) with small error.

$$r_{ss}(m) = \frac{1}{N} \sum_{n=-l}^{N-l} s(n)s(n+m)$$

(6)

$$r_{ss}^*(m) = \frac{1}{N} \sum_{n=1}^{N-1} s(n)s(n+|m|)$$

(7)

However, since (7) is aimed to be substituted into (3) and (4) finally, the term $\frac{1}{N}$ can then be cancelled at this point. We can rewrite (7) to (8).

$$r_{ss}^*(m) = \sum_{n=1}^{N-1} s(n)s(n+|m|)$$

(8)

which represents our first important realization that

$$r_{ss}^*(m) = r_{ss}^*(-m)$$

(9)

Experimenting with a real sound sequence of 50 blocks and based on the simplified equation in (8), we can calculate the value of $r_{ss}^*(0)$, $r_{ss}^*(1)$ and $r_{ss}^*(2)$ shown in Fig.2.

Using (12), we can rearrange the simplified coefficient equation $a_1$ and then $a_2$ to (13) and (14).

$$a_1 = \frac{(r_{ss}^*(0) - r_{ss}^*(2))}{2(r_{ss}^*(0) - r_{ss}^*(1))}$$

(13)

$$a_2 = 1 - a_1$$

(14)

Obviously, we can notice that the coefficient equations in (13), (14) and the autocorrelation in (8) are much simpler than those in (3), (4) and (5). From now on, we will use (8), (13) and (14) instead to implement the system. In order to confirm the correctness of the simplifications, we can calculate the modified coefficient $a_1$, $a_2$ in comparison to the original coefficient $a_1$, $a_2$ as shown in Fig.3 where the original coefficient $a_1$, $a_2$ are shown in solid lines while the modified $a_1'$, $a_2'$ are shown as dash lines. The plots prove that the simplified calculation method gives the values of $a_1'$, $a_2'$ in identical to the original $a_1$, $a_2$ and allow us to implement the adaptive predictor block much simpler than the original algorithm.

IV. PERFORMANCE OF THE MODIFIED ADPCM

After the correctness of the coefficient simplifications has been shown in Fig.3 in comparison to the coefficient values of the original algorithm, we can prove further the performance of the method by employing the algorithm into the ADPCM which is then denoted as a modified ADPCM. SNR of the modified ADPCM can be simulated on MATLAB and plotted in Fig.4 in comparison to the plots of the original ADPCM. It is seen that there is a good agreement between the SNR of the modified and the original algorithm. However, the difference of the SNR should be considered concurrently with the hardware size of both algorithms. The difference of the plots at the beginning block of Fig.4 are described that the signal power of the beginning of this block is very low or it is the area of quiet sound. Then the noise dominates in this region.

![Figure 3 Comparison plots of $a_1$, $a_2$, $a_1'$, $a_2'$](image)

![Figure 4 SNR of ADPCM vs. Modify ADPCM](image)

---

III-189
V. FPGA IMPLEMENTATION

This section shows statistical comparison of hardware usage between the conventional ADPCM and the proposed modified ADPCM on XILINX FPGA [6] XC4062XL-PC145. Table 6 shows that the gates usage of the proposed algorithm is reduced from 79.08% to 16.41%. The hardware size of the proposed modify ADPCM are 4.6 times smaller than the original ADPCM with a relatively the same SNR performance.

Table 6 Device utilization for 4062XL-PC145

<table>
<thead>
<tr>
<th>Resource</th>
<th>ADPCM</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLB FG Function Generators</td>
<td>3810</td>
<td>768</td>
</tr>
<tr>
<td>CLB H Function Generators</td>
<td>1952</td>
<td>300</td>
</tr>
<tr>
<td>CLB Flip Flops</td>
<td>3644</td>
<td>756</td>
</tr>
<tr>
<td>Use %</td>
<td>79.08%</td>
<td>16.41%</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

This paper proposes a modification of ADPCM algorithm to be practically suitable for hardware implementation of portable audio devices. We have shown that the most critical functional block of ADPCM are predictor coefficient calculation algorithm which requires a complicate correlation algorithm and resulting in high computation and hardware complexity. A mean to simplify the algorithm has been shown and proved that the complexity is greatly reduced while the sound quality measured as SNR remain relatively the same to those of the original ADPCM. The simplification results in a reduction of the chip area by 63% from the area required by the original algorithm hence the reduction of power consumption. Thus it is suitable for pocket solid-state speech and audio recorder. The simplification methods can also be extended to apply for other coding approaches such LPC.

VII. REFERENCES