Lab2 - lossless audio compression

The aim of this laboratory is to implement a system that performs lossless audio compression, to obtain this it has been implemented an audio compression algorithm, the codec is based on linear prediction and entropy coding of the prediction error. Different blocks have been implemented, see the scheme for reference:

INSERIRE FIGURA!!!!!

Predictor - it is an FIR filter that removes the correlation between samples, its effectiveness depends on its degree, it is useful because after that we have just to encode samples that cannot be predicted from past samples

Entropy encoder - it is a block that implements Rice-Golomb coding, this is a type of encoding that performs a lossless compression of data

Task 1

2 - Firstly we implemented three functions that compute the prediction error, one for polynomial predictor of first order, another for a second order predictor and a third function that can perform a generic prediction filtering, it accepts as an input coefficients for the filter, in particular, given coefficients implement a fifth order predictor, to obtain this we produced an FIR filter based on a circular buffer (as in the previous laboratory); easier predictors (first and second order) use just a linear buffer and fixed coefficients.

3 - Putting together predictors with reconstructing filters (matching their orders) we tested all the predictors we wrote, finding that the difference between the reconstructed signal and the input signal is null

4 – We implemented the entropy coder, it is based on Rice-Golomb coding, the algorithm computes the code associate to every sample: it is composed by a sign bit, the quotient of the division by a known constant (K) and the reminder of the division.

To store bits, we used the provided macro PACK\_BITS, it puts in a register the bitstream, then, when it’s full it stores it in memory.

5 – We verified that the codec works properly encoding some samples (using k = 9) and decoding them using a given decoder, then comparing the output of the decoder and the input of the encoder we found a null error, so the implementation of the decoder was correct.

6 – We used the codec to compress 8192 16-bit samples, so we had 1769472 bits in total to code, and we measured the number of encoded bits, the table below summarizes measurements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | No predictor | P1 | P2 | P5 |
| 5 |  | 165737 (high error) | 111755 (high error) | 93204 (high error) |
| 7 |  | 97869 (high error) | 84464 (error = 0) | 79941 (error = 0) |
| 9 | 141110 (error = 0) | 93526 (high error) | 90751 (error = 0) | 90230 (error = 0) |

We measured this parameter using different predictors, changing for each the K of the algorithm; it is possible to notice that, encoding directly the input samples, without using a predictor, for a low value of K (5 and 7), it is not possible to estimate the number of encoded bits since the buffer that should be used exceeds the memory that is actually allocated so the algorithm enters an infinite loop and it is not possible to evaluate this parameter.

As expected, using the same K, the number of packed bits decreases increasing the order of predictor.

7 – We measured also the number of clock cycles to perform the compression, using different levels of optimization, to obtain the number of clock cycles per sample it is sufficient to divide it by the number of samples processed (8192).

Without any optimization:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | No predictor | P1 | P2 | P5 |
| 5 |  | 10 | 5 | 4 |
| 7 |  | 4 | 29209 | 29218 |
| 9 | 41046 | 29460 | 29369 | 29458 |

Setting the optimization level at 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | No predictor | P1 | P2 | P5 |
| 5 |  | 26849 | 17100 | 16066 |
| 7 | 67681 | 15392 | 15396 | 15352 |
| 9 | 17875 | 15392 | 15396 | 15352 |

As expected, the number of clock cycles decreases if the optimization level increases.

8 – The maximum sampling frequency depends on CPU clock frequency and on the number of clock cycles per sample, so it depends on the type of predictor selected, the optimization level and K

Task 2

We used the developed system to implement a real-time audio coding, so we add to the given code the predictor and the encoder developed in the first part of the lab, the given decoder and the reconstructor.

2 - We counted the number of encoded bits for every channel and sampled this value every second exploiting the periodic interrupt generated by the RTC, so that we could estimate the number of bits processed per second, to make this estimation we set a fixed K (9), we choose the fifth order predictor, and we disabled compiler’s optimization. Earing whit earphones the output of the DSP card we noticed that lowering the sampling frequency the smoothness of the output audio increases, in fact is possible to notice that at lower frequencies the number of produced bits is high enough to maintain the sample rate asked by the DAC (equal to the one of the ADC).

|  |  |
| --- | --- |
| Sampling frequency (kHz) | Number of bits per second |
| 48 | 245147 |
| 24 | 224095 |
| 16 | 193493 |
| 12 | 148076 |

3 – Earing with earphones the output of the DSP we estimated the highest sampling frequency that permits not to degrade the quality of the signal, we found that the highest one is 24kHz, to obtain this frequency wee had to set the optimization level to the highest permitted level (3).

4 – We modified the main loop code to average the bitrate, to have a signal that is not too variable and fast we used a pop song, the 30 second average bitrate is reported, varying k and predictor we found that:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | No predictor | P1 | P2 | P5 |
| 5 |  | 180231 | 183258 | 222653 |
| 7 | 151149 | 223738 | 223830 | 223937 |
| 9 | 149122 | 268479 | 268479 | 268479 |

The uncompressed audio has a bitrate of 416000 bit/s, it is possible to notice that the bitrate of the compressed audio stream is further lower, in fact, compression is a technique used to achieve this aim; bitrate depends mainly on k, while it is a bit less affected by the choice of predictor, it is always possible to notice that without predictor, with low values of k, it is not possible to take measures on the algorithm.

5 – We used the provided function rice\_parameter to estimate the optimal value of K to perform the encoding, forcing the highest level of optimization and using a sampling frequency of 24kHz, we found that:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | No predictor | P1 | P2 | P5 |
| Estimated K | 6 | 6 | 6 | 6 |
| bitrate | 229458 | 201054 | 201990 | 203216 |

It is possible to notice that the estimated value of K is independent on the choice of predictor, in fact it is estimated starting from the samples, since we used always the same signal to test it, it does not change.

6 – We found that, more than on the kind of audio source, the value of K estimated by the function depends mainly on the intensity of the signal, in fact, using the same audio signal, we found that, to a high level of volume corresponds a high K (10, 11), and the bitrate is about 365000, while, lowering the volume to an intermediate value K goes to 8 (with a bitrate of 315000) and lowering to the minimum the volume, k goes to 5/6 and the bitrate goes to 266000.