Appeared in the Proceedings of ICSPAT'98, Toronto, September 1998

Data Dependency in MPEG Audio Frame Decoding Time on a DSP Accelerator*

Robert Baumgartl, Kai Stuhlemmer, Hermann Härtig

Department of Computer Science
Dresden University of Technology, Germany
{robert.baumgartl,kai.stuhlemmer,hermann.haertig}@inf.tu-dresden.de

Abstract

One way to relieve a system's CPU from computationally-intensive multimedia applications is to incorporate a DSP as multimedia coprocessor into the system [2]. To generate efficient schedules for the multitasking DSP it is necessary to determine exact execution timing for complex multimedia algorithms. As a case study, this paper focuses on the timing of a DSP-based MPEG audio decoder and analyzes influence factors as layer, target bit rate and concurring DMA processes. We demonstrate that we are able to save approximately 60% of DSP processing time with scheduling deadlines based on our exact timing analysis.

1 Introduction

Stream-based multimedia algorithms usually require cyclic scheduling for which a deterministic execution time is a prerequisite. Unfortunately, many multimedia algorithms (e.g., MPEG Audio [1]) are very complex and, even worse, the complexity depends on the actual input data. Therefore, the determination of exact execution timing for encoding and decoding multimedia streams has to be done with a fine granularity which is impractical for complexity reasons. To overcome this dilemma, several approaches exist:

1. Use the CPU for processing multimedia streams and reserve resources generously, that is, choose a sufficient late deadline for multimedia streams.
2. Use the CPU for processing multimedia streams and allow deadline misses of multimedia applications. If very high quality is desired, no other CPU-intensive process may run.
3. Employ specialized hardware for the multimedia algorithm (e.g., MPEG chipsets) and use the CPU only for transporting data to and from the accelerator.

Solution 1 can guarantee a certain quality-of-service but is a waste of CPU resources. Solution 2 leads to a potential quality-of-service degradation which is often undesirable. Solution 3 gives guarantees and relieves the CPU but is potentially inflexible, because newly deployed multimedia algorithms require new specialized hardware.

Therefore, the goal of our project is to avoid these disadvantages by using a general-purpose DSP to accelerate several multimedia applications running simultaneously at the host [2]. The reasons for choosing a DSP as accelerator are:

- DSPs are flexible,
- DSPs are very powerful, their architecture renders them them ideal for multimedia algorithms,
- DSPs are comparatively cheap,
- the determination of DSP execution time is easy. Therefore, they can be operated with a high load under real-time conditions.

The simultaneity of DSP tasks requires cyclic scheduling within the DSP and, to facilitate the enormous performance potential of actual DSP

*This work was supported by the DFG with the Graduiertenkolleg "Tools for the effective Use of Parallel and Distributed Computing Systems" and the SFB 358.

types, the determination of very accurate process deadlines.

As a case study, this paper focuses on MPEG 1 audio decoding [3] as a complex multimedia algorithm example for a DSP architecture. The basis of the work is our implementation of a highly-optimized MPEG Audio decoder for the TMS320C44 DSP [4]. We study the timing behaviour of the decoder for audio streams with different contents, among them classical and pop music, natural noises, speech and some synthetic signals. Furthermore we demonstrate the influence of certain parameters such as the used layer, the target bit rate or the existence of concurring DMA processes. We demonstrate that a large amount of DSP processing time can be saved by obtaining exact decoder deadlines on a stream-by-stream basis without any quality-of-service degradation.

For a comparison with a general-purpose processor implementation we reiterated some of the measurements with a PC-based MPEG audio decoder.

2 Experimental Environment

All experiments were done on a 166 MHz Pentium MMX PC equipped with 64 MBytes of RAM and 512 KBytes 2nd level cache. As operating system of the host Linux was used. The DSP subsystem resides on a PCI plug-in card and comprises a TMS 320C44 DSP clocked at 50 MHz, 128 KWords of each global and local static memory, and an AMCC S5933 PCI controller.

The DSP decoder processes layer I and layer II MPEG 1 audio streams [4]. It works under real-time conditions and is able to decode all target bit rates required by the ISO standard [1]. All aural modes are supported. The decoder implementation is highly optimized and was done in assembler language only. To obtain exact execution timing we instrumented the code with DSP timer accesses which yield a timing measurement resolution of 2 instruction cycles.

The streams to be analyzed were grabbed from audio and test CDs, resulting in an original sampling rate of 44.1 kHz. Table 1 lists all examined streams. We converted the streams into MPEG audio format using the public-domain encoder of the ISO MPEG Audio Subgroup Software Simulation Group[1] running under Linux with differing encoding parameters.

[1] available from ftp://ftp.iuma.com/audio_utils/converters/source/

Name	length [s]	Characteristic
O Fortuna	120	classical music, highly dynamic
Summer knows	120	guitar, female voice, quiet
Mystic's Dream	120	choir, female voice, quiet
Disp Heroes	120	Heavy Metal, fast, loud
Rockin'	120	Rock music
Speech	60	male speech, movie excerpt
White Noise	35	analytical signal
Pink Noise	65	analytical signal
Shore	69	swelling and decaying surf
Zero	30	digital zero
Birds	75	tweeting birds

Table 1: Characteristics of analyzed audio streams

3 Results

3.1 Layer I Decoding

We begin our discussion with MPEG audio layer I. The streams were encoded in stereo mode with a target bit rate of 128 Kbps. The encoded streams from table 1 were decoded using our optimized DSP decoder. We analyzed the execution time of the decoder on a frame-by-frame basis for every stream and obtained relevant statistical parameters. Figure 1 shows the minimum, the maximum and the arithmetical average of the frame decoding times.

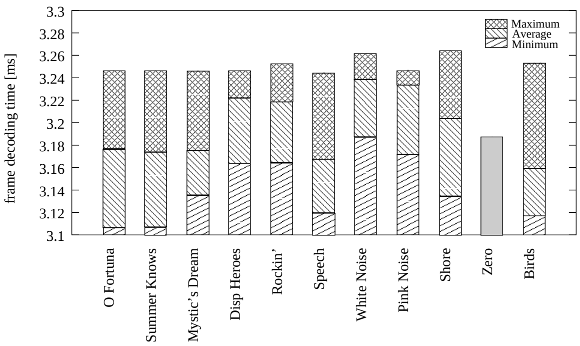

Figure 1: Frame decoding time parameters for layer I

The overall decoding time differences are comparatively small, especially when comparing music streams only. Synthetic noise has somewhat higher processing requirements, but the difference may be negligible. As expected, the frame decoding time variation of the "zero" stream is zero. The difference between the absolute minimum and maximum is 0.155 milliseconds (3884 DSP clock cycles).

Furthermore, we generated histograms (cf. figures 2 and 3) of the frame decoding time distribution which reveal the following:

- There are a finite number of peaks in the distribution resulting from the different number of encoded frequency bands.
- Louder, more dynamical music has a slightly differing distribution in comparison to quiet music.
- synthetic signals and noise have a more compact distribution with higher peaks resulting in a smaller decoding time variation.

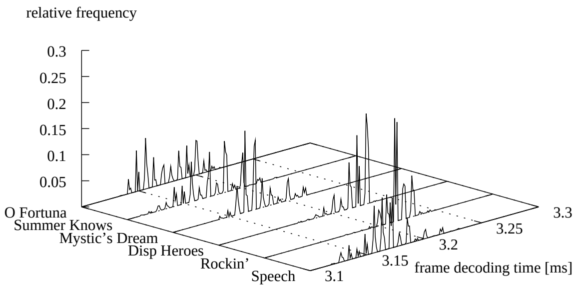

Figure 2: Distribution of frame decoding time for layer I (part 1)

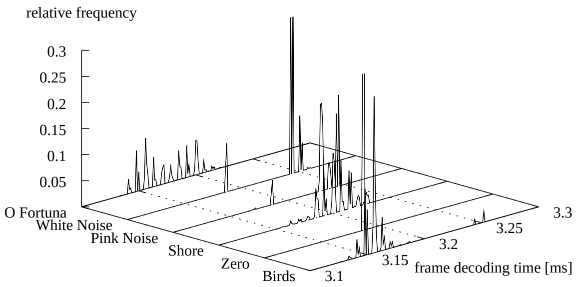

Figure 3: Distribution of frame decoding time for layer I (part 2)

The small frame decoding time differences, the short time needed to decode a layer I frame and the inevitable overhead render the exploitation of decoding time differences on a frame-per-frame basis unrealistic. Instead, the deadline of the MPEG decoder should be determined by the maximum frame decoding time of the stream, which can be obtained by a single run of our analyzer program.

In our example a sampling frequency of 44.1 kHz and a frame length of 384 words (layer I) is used. Therefore, the latest possible decoder deadline is 8.707 milliseconds. Reducing the deadline to the maximum frame decoding time of the stream (e.g., 3.246 milliseconds for the stream "O fortuna") saves approximately 60% of DSP processing time without affecting the quality of the decoded stream.

3.2 Layer II Decoding

In comparison to layer I, layer II adds more encoding complexity resulting in better audio quality or lower stream bit rates. Figure 4 shows the relevant statistical parameters of the frame decoding times of our streams encoded with MPEG layer II. Again, all streams were encoded in stereo mode with a target bit rate of 128 Kbps.

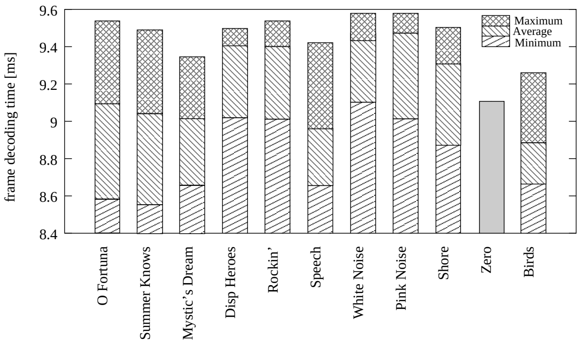

Figure 4: Frame decoding time parameters for layer II

In contrast to layer I, the frame decoding time varies more and the overall processing time requirements are nearly 3 times as high. The following can be observed:

- Synthetic noise has the highest processing time requirements. In comparison, it has the highest overall maximum frame decoding time as well as the highest minimum decoding time. This result was somewhat expected, because noise has frequency components throughout the audible spectrum. Shore noise behaves very similar to synthetic noise.
- The decoding timing does not necessarily depend on the character of the music. For instance, "O Fortuna" has a very high dynamic range, whereas "Summer knows" is very quiet, but both have almost identical statistical parameters. On the other hand, "Disposable Heroes" by Metallica is also very loud and very dynamically, but its parameters are much more similar to noise.

- With respect to the observed parameters, speech can not be distinguished from music.
- The difference between the most and the least complex audio frame is approximately 1.0 millisecond (25000 DSP clock cycles) resulting in a frame decoding time variation of about 10%.

Figures 5 and 6 show the histograms of the frame decoding time.

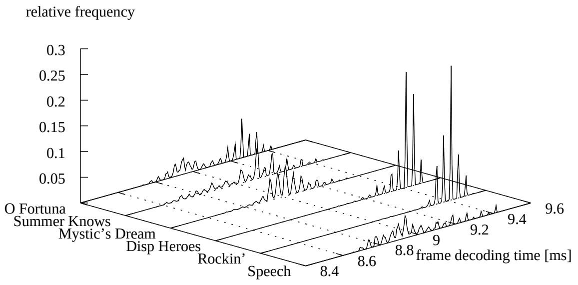

Figure 5: Distribution of frame decoding time for layer II (part 1)

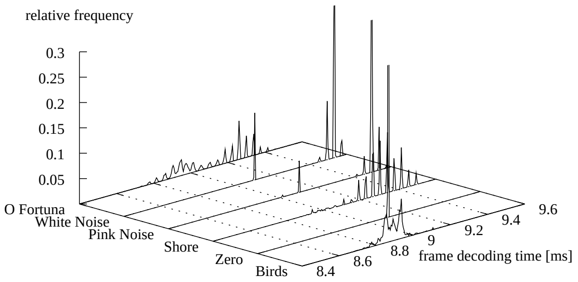

Figure 6: Distribution of frame decoding time for layer II (part 2)

The distribution of processing time for layer II is very similar to the distribution for layer I. Loud, dynamic music (e.g., "Disposable Heroes" and "Rockin"') has a more compact spectrum with higher peaks, which indicates that these streams have a lower variation in decoding time than quiet music. The highest variation in frame decoding time can be observed for speech.

The latest possible deadline for the decoder in layer II mode is 26.123 milliseconds (1152 words frame length, 44.1 kHz sampling frequency). Reducing the deadline to the maximum of the stream (e.g., 9.532 milliseconds for "O Fortuna") again results in approximately 60% processor time save. It even seems feasible to adapt the deadline of the decoder when a group of frames with significant lower processing requirements is, but this is beyond the scope of this paper.

3.3 Influence of Target Bit Rate

To get an overview of the processing requirements of the algorithm for different target bit rates we encoded a selected stream ("O fortuna") with different bit rates in layer II and obtained the statistical parameters (cf. figure 7).

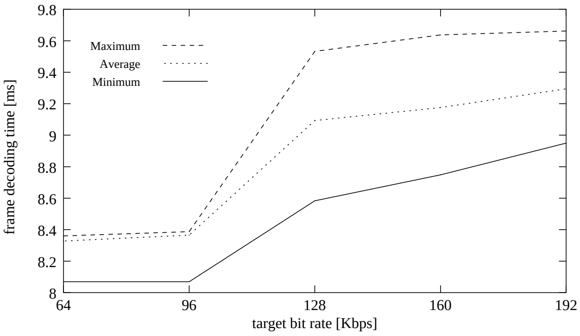

Figure 7: Statistical parameters for different target bit rates

As can be seen, the processing requirements do not increase linearly with the target bit rate. Instead, for low bit rates, the frame decoding time remains almost constant. Between 96 and 128 Kbps it raises considerable (the variation increases also), whereas for higher bit rates the increase slows.

We can conclude that as a first approximation it could suffice to differentiate between two decoder scheduling deadlines: One for the lower bit rates of 64 and 96 Kbps and one for the high bit rates between 128 and 192 Kbps.

For a sampling frequency of 44.1 kHz and a frame length of 1152 words for layer II the latest possible scheduling deadline for the decoder is 26.123 milliseconds. Choosing the respective maximum frame decoding time of the stream as decoder scheduling deadline saves between 60.0% (192 Kbps) and 65.3% (64 Kbps) DSP execution time.

3.4 Concurring DMA Transfers

As a last influence factor on DSP execution timing we investigated concurring DMA processes. To this aim, two independent DMA processes which were responsible for data transfer from and to local memory were added to the DSP software.

The amount of transferred data met exactly the amount of input and output data of the MPEG audio algorithm. This extension did not have any visible effect on execution timing of the running MPEG audio decoder.

3.5 PC Decoder

To compare the timing behaviour of the DSP implementation with a general-purpose processor we reiterated some experiments with a instrumented version of the Linux-based MPEG audio decoder `maplay`[2]. Again, we analyzed the layer-II-encoded streams. During the measurements, all other system activity was reduced to a minimum. The results are depicted in figure 8.

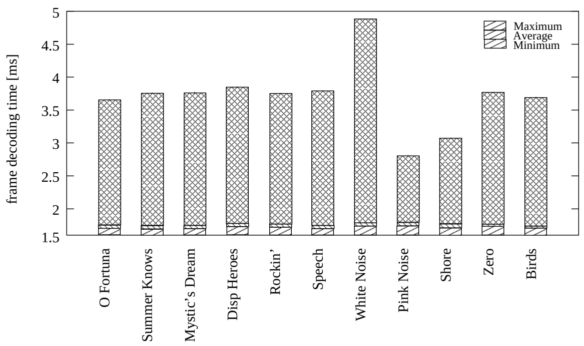

Figure 8: Frame decoding time parameters (layer II) for PC decoder

In comparison to the DSP implementation, the variation is much higher. The maximum frame decoding time is two to three times as high as the average. The average is very close to the minimum frame decoding time indicating relatively few frames with excessive timing requirements (for instance caused by cache effects). Scheduling this MPEG audio decoder therefore means either wasting CPU resources or potential quality of service degradation by allowing occasional deadline misses.

4 Conclusions & Outlook

Briefly, our experiments indicate the following:

- For layer I, streams do not differ significantly in their frame decoding time distribution. On the other hand, the differences between individual streams are much more significant for layer II.
- By choosing the maximum frame decoding time as decoder scheduling deadline, with our configuration it is possible to save up to 65% of DSP processing time without quality degradation in comparison to donating all DSP resources to the decoder.
- There is no rule-of-thumb for the estimation of the complexity of a given music stream (e.g., very dynamic music does not result in a exceptionally high decoding time).
- Speech does not differ significantly from music.
- Parallel DMA processes don't influence the decoder timing in our environment.
- Stream decoding using an Intel Pentium processor results in significantly higher frame decoding time variation .

The next step of our project is to analyze a MPEG audio encoder the same way and then run both encoder and decoder as accelerator tasks using a DSP real-time kernel with very accurate deadlines.

[2] available from `ftp://ftp.cs.tu-berlin.de:/pub/multimedia/maplay1.2`

References

[1] ISO/IEC, *IS 11172-3, Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s*, Part 3: Audio, 1995

[2] Robert Baumgartl, Hermann Härtig, *On the Integration of DSP Hardware into a Microkernel-based Operating System*, Proceedings of ICSPAT'96, Boston, 1996

[3] Davis Pan: *A Tutorial on MPEG / Audio Compression*, IEEE Multimedia Journal, Summer 1995

[4] Kai Stuhlemmer: *Audio-MPEG-Decoder for DSP*, Term paper, Dresden University of Technology, 1997 (in German)