

Worst-Case Execution Times Analysis of MPEG-2 Decoding

Peter Altenbernd
C-LAB
peter@c-lab.de

Lars-Olof Burchard
PC²
33095 Paderborn, GERMANY
baron@upb.de

Friedhelm Stappert
C-LAB
fst@c-lab.de

Abstract

This work presents the first worst-case execution times (WCET) analysis of MPEG-decoding. Solutions for two scenarios (Video-on-Demand (VoD), live) are presented, serving as examples for a variety of real-world applications. A significant reduction of overestimations (down to 17 %, including overhead) during WCET analysis of the live scenario can be achieved by using our new two-phase decoder with built-in WCET analysis that can be universally applied. It is even possible to predict the exact execution times in the VoD scenario.

This work is motivated by the fact that media streaming service providers are under great pressure to fulfil the quality-of-service promised to their customers, preferably in an efficient way.

Keywords: WCET Analysis, Media Streaming, MPEG Decoding, QoS

1. Introduction

Quality-of-service (QoS) and resource reservation play an important role for *audio/video (AV) streaming*. AV streaming applications are used for *video-on-demand, TV broadcasting and live* scenarios, e.g. video conferencing. An example for a AV streaming format is *MPEG-2*¹ [10], a popular and wide-spread standard for video and audio compression. The MPEG standard covers video and audio of various sizes and qualities. QoS guarantees allow to meet exactly the requirements for the playback of the delivered streams without loss of quality, e.g. packet loss or congestions during playback.

In general, scheduling algorithms concerned with dispatching several streams (as needed for conferencing, or for playing MPEG-4 [11]) on client systems, usually do

¹In the following sections, we will refer to the MPEG-2 standard, although the presented decoding methods and implementations can also be applied to MPEG-1 streams.

not consider timing aspects. This means some "best-effort" strategy is chosen without considering necessary and available computing and network resources or QoS guarantees for the decoding process. However, though AV streaming can be considered as a soft real-time problem in which deadline misses do not cause catastrophic failure, service providers are under hard pressure to guarantee the QoS promised to their customers, in order to avoid litigations.

This has lead to using *fixed-bandwidth allocation* or *real-time scheduling* allowing guaranteed QoS by taking properties of the streams into account. However, fixed-bandwidth allocation lacks the flexibility of reacting to changes during the streaming process (e.g. caused by variable bitrate streams), and results in wasting resources since the peak bandwidth always has to be allocated. In contrast to this, real-time (i.e. time-based, priority-driven) scheduling has the advantage of a more efficient usage of the available computing and network resources [14]. Saving computing resources can allow to implement cheaper designs and enables the CPU to perform additional tasks.

Worst-case execution times analysis (WCET) is the foundation for using real-time scheduling. The problem to overcome in order to achieve "good" results, is to minimize the overestimation during WCET analysis. Especially, when using variable bitrates during encoding (for achieving the highest possible compression), the decoding times of the pictures in these streams have large differences, thus may result in enormous overestimations when using simple WCET techniques. However, in this document we show methods for two scenarios (live, VoD) trying to reduce the gap between predicted worst-case and the actual execution times. To the authors' knowledge this is the first paper addressing WCET analysis of MPEG decoding.

In the VoD scenario, an off-line computation of the WCET allows to provide the exact figures to the scheduler, by simply parsing the stream. Further we show, how to group the collected WCET information and store them in MPEG headers.

For the live scenario, we integrated the WCET analysis into the decoder (thus partly taking input data into account),

in contrast to a naive method, i.e. simply extracting the theoretically possible worst-case (thus ignoring all input data). This is achieved by two ways: Firstly, we analysed parts of the control flow of the decoding process, resulting in safe assumptions about relations between certain input data and the corresponding path taken in the control flow. Secondly, WCET information is collected in an early phase of the decoding, in order to make predictions about the execution time of a later phase.

Since our focus was on the structure of the decoding process, the WCET analysis was made on source code level only, counting basic operations like additions, multiplications, etc.. For a complete WCET calculation the results of this document must be combined with a low-level (i.e. hardware related) analysis of the decoding process. For a number of sample streams in the live scenario, the additional resource requirements, i.e. overhead together with the overestimation, could be decreased to an average of only about 17 % (85 % resource utilization) compared to more than 160 % (39 % resource utilization) using the naive approach (see Section 4).

In this document, after discussing related work, we describe important parts of the MPEG format and their relation to the proposed solutions. The following sections deal with a VoD scenario solution (Section 3.2), and the universal two-phase decoder presented in the live scenario (Section 3.3), respectively. The results of our experimental evaluation are shown in Section 4. Finally, Section 5 gives some conclusions.

2. Related work

[4] describes a best-effort method for predicting execution times of MPEG-decoding by using information from previously decoded pictures and the length of the MPEG-encoded pictures, i.e. the storage space they require. The results in that document were based upon the examination of six MPEG streams. Although rather successful for five of these streams, the document also shows limitations of the method. In particular, it is possible to have underestimations and moreover it is unclear whether it is possible to use such an approach for other than the examined streams. The approach presented in this paper seems to be the first using WCET analysis for MPEG decoding. Due to the different environments of the two approaches, [4] uses measurements of the required clock cycles without considering the decoder's source code in any way. Because the solutions in this paper concentrate on high-level examinations, they cannot be directly compared.

Trying to measure WCETs [7, 3], is known to be too unsafe and inaccurate. Hence, the goal is to have an analytical method, like this approach. Due to complexity reasons an exact analysis is usually intractable, so most approaches try

to approximate the WCET. This has always to be done in a way that does not underestimate at all, but reduces overestimations to the essential necessary part, as in this case.

Most work in the field of WCET either focuses on the high (i.e. programming code) level [2, 6, 8], or on the low (i.e. hardware) level [9, 13]. *High-level methods* usually try to predict timing with respect to mutually depending program parts that could exclude each other, in a given program to be analysed. The approach in this paper can be classified to the same level, but in contrast to these methods, the behaviour of the program (the MPEG decoder) is known from the very beginning. Hence, this paper does not focus on finding out the dependencies within the program, but on the data given to the program, and its effect on the WCET.

At the *low level*, usually the caching and pipelining behaviour and its impact on the WCET is analyzed [15, 16, 5]. Although, this of course is also relevant to MPEG decoding, this paper rather emphasizes the higher level: it is assumed that known low-level techniques can be connected to the implementations of this paper to eventually predict the WCET.

3. WCET of MPEG decoding

As explained above, we will now describe the MPEG format and the different scenarios.

3.1. MPEG

A complete description of the MPEG compression scheme is far beyond the scope of this document (for details, see [10]), therefore we will concentrate on a short explanation of the coding steps important for our WCET analysis.

In general, compression methods of MPEG can be divided in two parts: one compresses data on a picture level (i.e. the JPEG compression scheme), the other uses similarities between pictures for data reduction. This process is called *motion compensation*. There are 3 different picture types, distinguished by the type of motion compensation used during the encoding process: *I*-, *P*- and *B*-pictures. *I*-pictures are just JPEG-compressed pictures which contain no references to other pictures of the stream, *P*-pictures contain references to preceding, *B*-pictures contain references to pre- and succeeding pictures. Each picture is divided into smaller units: *slices*, *macroblocks*, and *blocks*. Slices contain several macroblocks and are used for error control: some parameters of macroblocks are restored with each start of a new slice. This allows to decode a picture even in case a single slice is missing, e.g. due to loss of packets on the network. Macroblocks are units representing 16x16 pixels. On the macroblock level, the motion compensation process is applied, i.e. each macroblock in *P* and *B* pictures can be replaced by a vector. The problem for the WCET analysis in this case is, that only the upper limit of

macroblocks described as references is known in advance. Blocks are the smallest units in MPEG pictures. To these units, the *discrete-cosine-transform* (DCT) [1], a function similar to the Fourier transform, is applied.

The most important (JPEG-like) compression methods for our considerations are *variable-length coding* (VLC, similar to huffman coding) and the *discrete cosine transform* (DCT). During the last encoding step, the MPEG encoder uses variable-length codes for further compression of all the results (i.e. motion vectors, DCT coefficients) of the previous steps. Our two-phase decoder described in Section 3.3 takes advantage of the fact that the variable-length decoding is independent from the succeeding, most expensive/time consuming steps of the decoding process such as the inverse DCT.

3.2. WCET analysis in a Video-on-Demand scenario

An example for a VoD system could be a hotel application which provides access to videos on a server in each hotel room via television and settop-boxes. In the VoD scenario, streams can be examined and even modified before delivery. Hence, an exact WCET calculation can be done off-line and in advance. The only remaining problem is to group all available information in a reasonable way into MPEG headers in order to be used by the end system's scheduler.

The solution described here is based on precomputed execution times delivered within the MPEG stream and evaluated by the decoder in order to provide exact WCET information and deadlines for the scheduler on the client system. Having control over the streams on the server makes it possible to examine each stream, calculate the necessary computing time (with a decoder-like tool) and store this information as *user data* within the stream. User-data is defined by the MPEG standard in order to place additional data (i.e. data not related to the decoding process) within the stream. It can be stored in each picture header, implemented by using escape sequences. In our case, we chose the picture header (Fig. 1), which allows to cover sequences of arbitrary length, for instance information about a longer sequence of pictures can be stored in the user data field of each I picture.



Figure 1. User-data fields in a stream

The usual behaviour of an MPEG-2 decoder is to ignore user data, which means in case real-time scheduling is not required (a control screen might not need this functionality), any standard MPEG-2 decoder can be used to playback the

modified streams without being influenced by the additional data. Although depending on the actual amount and type of WCET related data, the overhead caused by the additional user data in the stream is negligible.

The connection to the low-level WCET analysis can be implemented in two ways: Firstly, the WCETs for a number of client architectures can as well be stored as user data in the MPEG streams (at the server's side). Secondly, in a heterogeneous environment with many different clients, it is possible to perform the low-level analysis online at the client's side.

3.3. WCET analysis in a live-video scenario

The strategy described in the previous section is not feasible for the live-video scenario, e.g. a video conference with several participants. The encoded videos are delivered online and a complete preanalysis is impossible.

In the new approach described in this section, a complete MPEG encoded picture contained in a given stream is examined in order to collect information about the WCET of the decoding process. In order to keep the computing overhead small, the collecting process is integrated in the decoder. This allows to reuse the decoded parts of the picture for subsequent decoding steps.

In general, MPEG uses two types of algorithms for the compression of single pictures. The first (the VLC compression) can be considered as a general purpose algorithm, while the others are used to compress picture data (e.g. DCT, motion compensation, see Section 3.1). Both types of algorithms are usually run in one phase, completely decoding each macroblock before advancing to the next.

However, in our new approach, this process was restructured into two phases corresponding to the two types of algorithms: The first phase decodes all VLCs, the second phase decodes the actual picture information (see Fig. 2). The WCET of the first phase is predicted during a preprocessing step: the size of the encoded picture is used to determine the maximum amount of VLCs contained in the picture which then can be used to estimate the corresponding WCET. During phase one, it is possible to identify the whole structure of the picture. This information can be used to accurately predict the WCET of the second phase, which decompresses the actual picture data. The evaluation (see Section 4) shows the success of this procedure.

All necessary processing steps are further described in the following.

3.3.1 Preprocessing

The most difficult part of the two-phase decoder is the preprocessing, i.e. predicting the WCET of the first phase. The first phase can be divided in the following parts:

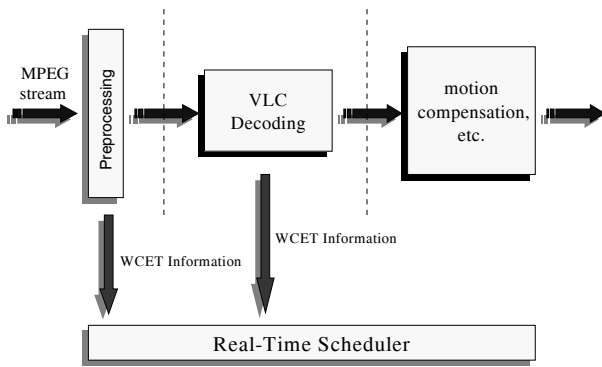


Figure 2. Two-phase decoder

1. slice decoding (SLICE), i.e. slice related parameters
2. macroblock decoding (MB), i.e. macroblock type etc.
3. motion vector decoding (MV), in P and B pictures only
4. intra / non-intra block decoding (IBLOCK / NIBLOCK) in I / P and B pictures, i.e. decoding the DCT coefficients
5. miscellaneous (MISC)

The DCT coefficients (IBLOCK, NIBLOCK) require most of the space of a picture and also most of the decoding time (Part 4). The decoding time of the other components (SLICE, MB, MV, and MISC) is much smaller (parts 1, 2, 3, 5) and upper bounds on their WCETs can easily be found. Therefore, the strategy of reducing the overestimation of the whole first phase is based on determining the number of non-zero DCT coefficients encoded in a picture.

Depending on the picture type, the actual WCET differs. In this paper, the procedure for estimating I pictures is described. P and B pictures require equivalent computations. The main difference is the presence of motion vectors (MV) in P and B pictures. In the following, the strategy for determining the number of non-zero DCT coefficients in a given I picture is presented.

Blocks in I pictures (see Section 3.1) are structured as follows: the first coefficient in the block (called *DC coefficient*) must always be present, it is encoded differently from the subsequent coefficients (called *AC coefficients*). The maximal number of non-zero coefficients in a block is 64, which means there are at most 63 AC coefficients. Each coefficient (DC or AC) is represented using either a predefined VLC or the escape sequence. The simplified source-code of the function that decodes intra-coded blocks is shown in Fig. 3. It shows the structure of the function and the different basic blocks (denoted $B_1 - B_{13}$):

```
void decode_intra_block()
{
  B1(); /* initialization */

  B2(); /* decode DC coefficient */

  /* decode AC coefficients */
  while(1)
  { /* loop executed at most 63 times */

    B3(); /* common to all VLCs */

    /* identification of the coefficient */
    if (length<=6)
      B4(); /* shortest VLCs, 3-6 bit */
    else if (length==6)
      B5(); /* escape sequence and 6 bit VLCs */
    else if (length==7)
      B6(); /* 7 bit VLCs */
    else if (length==8)
      B7(); /* 8 bit VLCs */
    else if (length==9)
      B8(); /* 9 bit VLCs */
    else if (length==10)
      B9(); /* 10 bit VLCs */
    else if (length==11)
      B10(); /* 11 bit VLCs */
    else if (length>11)
      B11(); /* 12 - 17 bit VLCs */

    /* decode run/level */
    if (escape)
      B12(); /* escape sequence */
    else
      B13(); /* all other coefficients */

    if (end)
      return;

    /* advance to the next coefficient */
    FlushBuffer();
  }
}
```

Figure 3. Intra-coded block decoding (simplified)

| | |
|----------------|--|
| B_1 | initialization |
| B_2 | decoding the DC coefficient |
| loop: B_3 | executed for each AC coefficient |
| $B_4 - B_{11}$ | decode AC coefficient, B_5 decodes both 6 bit predefined VLCs and escape sequences |
| B_{12} | postprocessing of escape sequences |
| B_{13} | postprocessing of predefined VLCs |

The VLCs of DC and AC coefficients have different lengths (predefined VLCs require between 3 and 17 bits, escape sequences always require 24 bits), and also require different decoding times. Hence, for the WCET calculation of the first phase it is not sufficient to determine the shortest pos-

sible VLC of an AC coefficient but the length of the VLC which requires the most decoding effort per bit. In the following, this particular VLC is denoted VLC_{max} .

| Basic Block | WCET (C_i) |
|-------------|---|
| B_1 | $C_1 = (1, 0, 10, 1, 1, 1) \cdot c_{cycles}$ |
| B_2 | $C_2 = (1, 0, 3, 1, 0, 0) \cdot c_{cycles}$ |
| B_3 | $C_3 = (0, 0, 20, 7, 0, 0) \cdot c_{cycles}$ |
| B_4 | $C_4 = (1, 0, 3, 0, 0, 1) \cdot c_{cycles}$ |
| B_5 | $C_5 = (1, 0, 5, 2, 0, 1) \cdot c_{cycles}$ |
| B_6 | $C_6 = (1, 0, 6, 2, 0, 1) \cdot c_{cycles}$ |
| B_7 | $C_7 = (1, 0, 6, 3, 0, 1) \cdot c_{cycles}$ |
| B_8 | $C_8 = (1, 0, 7, 4, 0, 1) \cdot c_{cycles}$ |
| B_9 | $C_9 = (1, 0, 8, 5, 0, 1) \cdot c_{cycles}$ |
| B_{10} | $C_{10} = (1, 0, 9, 6, 0, 1) \cdot c_{cycles}$ |
| B_{11} | $C_{11} = (1, 0, 10, 7, 0, 0) \cdot c_{cycles}$ |
| B_{12} | $C_{12} = (1, 0, 8, 3, 1, 0) \cdot c_{cycles}$ |
| B_{13} | $C_{13} = (1, 0, 8, 0, 0, 0) \cdot c_{cycles}$ |

Table 1. WCET of the basic blocks

The number of operations required by each basic block is shown in Table 1 (including effort for control structures). To represent the number of operations in a basic block, vectors with six components are used, each representing one of the six different operations:

(< ADD >, < MULT >, < MEM >, < CMP >, < BIT >, < SHIFT >)

This notation allows to easily use the WCET in equations. For the calculation of the required clock cycles in each basic block ($C_1 - C_{13}$), the vectors are multiplied with the variable c_{cycles} which represents the actual number of clock cycles required by each operation. For example, when using the PowerPC 604 [12] this means $c_{cycles} = \langle (2, 3, 20, 3, 1, 1) \rangle$.

For encoding the coefficients, two types of VLCs can be used: predefined VLCs and VLCs, which start with the escape sequence and require 24 bits (see Section 3.1). Let $l_i, i \in \{4, \dots, 11\}$, denote the lengths (in bits) of the predefined VLCs, decoded by B_i (i.e. $l_4 := 3, l_5 := 6, \dots, l_{11} := 12$), and l_{12} denote the length of VLCs starting with the escape sequence (i.e. $l_{12} := 24$). The decoding time per bit required for each VLC expressed by

$$\frac{C_i + C_3 + C_{13}}{l_i}, i \in \{4, \dots, 11\}$$

for predefined VLCs and

$$\frac{C_5 + C_3 + C_{12}}{l_{12}} = \frac{C_5 + C_3 + C_{12}}{24}$$

for escape sequences.

Let $B_v, v \in \{4, \dots, 11\}$ denote the basic block which decodes VLC_{max} , C_v the corresponding WCET, and let $l_w, w \in \{4, \dots, 12\}$ denote the length of the corresponding

coefficient². v and l_w are computed, using the following equation, which determines the WCET per bit required for decoding VLC_{max} :

$$\begin{aligned} & \max \left\{ \underbrace{\left\{ \frac{C_i + C_3 + C_{13}}{l_i} \mid i \in \{4, \dots, 11\} \right\}}_{\text{predefined VLCs}} \cup \underbrace{\left\{ \frac{C_5 + C_3 + C_{12}}{l_{12}} \right\}}_{\text{escape sequence}} \right\} \\ &= \max \left\{ \frac{C_4 + C_3 + C_{13}}{3}, \frac{C_5 + C_3 + C_{13}}{6}, \frac{C_6 + C_3 + C_{13}}{7}, \right. \\ & \quad \frac{C_7 + C_3 + C_{13}}{8}, \frac{C_8 + C_3 + C_{13}}{9}, \frac{C_9 + C_3 + C_{13}}{10}, \\ & \quad \left. \frac{C_{10} + C_3 + C_{13}}{11}, \frac{C_{11} + C_3 + C_{13}}{12}, \frac{C_5 + C_3 + C_{12}}{24} \right\} \\ &= \frac{C_v + C_3 + C_k}{l_w} \end{aligned} \quad (1)$$

with $k = 12$ (if $l_w = 24$) or $k = 13$ (if $l_w \neq 24$). The actual values of v and l_w depend on the value of the variable vector c_{cycles} .

Let l denote the size (in bits) required by a given I picture, and $n_{IBlocks}$ the number of intra coded blocks in this I picture ($n_{IBlocks}$ also equals the number of DC coefficients in the picture). DC coefficients require at least 3 bits, therefore the minimal space required by the DC coefficients equals $3 \cdot n_{IBlocks}$. This means, the maximum number of non-zero AC coefficients equals $\left\lfloor \frac{l - 3 \cdot n_{IBlocks}}{l_w} \right\rfloor$. Let $C_{\{SLICE, MB, MISC\}}$ denote the WCET of the remaining parts of the first decoding phase as described at the beginning of this section (C_{MV} is not included because motion vectors only appear in P and B pictures). These variables depend not only on the vector c_{cycles} but also on the resolution and the chrominance (i.e. colour) format of a given stream.

The function for estimating the WCET of the complete first decoding phase (C_{Phase1}) can be computed as follows:

$$\begin{aligned} C_{Phase1}(l) &:= \underbrace{n_{IBlocks} \cdot (C_1 + 4 \cdot C_{SLICE} + C_{MB} + C_{MISC})}_{\text{initialization and DC decoding}} \\ &+ \underbrace{\left\lfloor \frac{l - 3 \cdot n_{IBlocks}}{l_w} \right\rfloor}_{\text{space required by DC coeff.}} \cdot \underbrace{(C_v + C_3 + C_k)}_{\text{WCET for decoding } VLC_{max}} \end{aligned} \quad (2)$$

The function $C_{Phase1}(l)$ is used for the preprocessing in the two-phase decoder. It is a function of the length (in bits) of an I picture. Since DC coefficients are always present in I pictures, the expression $l - 3 \cdot n_{IBlocks}$ cannot be less than zero.

Overestimation The overestimation, i.e. the largest possible error of the preprocessing function, can be calculated equivalently to the procedure in the previous section. For

²Since B_5 decodes both 6 bit predefined VLCs and VLC starting with the escape sequence, different indices (v and $w, v \in \{4, \dots, 11\}$ and $w \in \{4, \dots, 12\}$) are chosen.

the purpose of simplification, the execution times of SLICE, MB, and MISC were not taken into account. However, this does not influence the correctness of the equations.

Let VLC_{min} denote the AC coefficient requiring the least decoding effort per bit, and let $B_{v'}$ denote the basic block which decodes VLC_{min} , $C_{v'}$ the corresponding WCET, and let $l_{w'}$ denote the length of the corresponding coefficient, $v' \in \{4, \dots, 11\}$, $w' \in \{4, \dots, 12\}$. Let l_i , $i \in \{4, \dots, 11\}$, again denote the lengths of the AC coefficients decoded by B_i (let $l_{11} := 17$, since B_{11} decodes also 17 bit coefficients and in this case, the minimum is computed), and $l_{12} := 24$. v' and $l_{w'}$ can be computed analogous to Equation 1:

$$\begin{aligned} & \min \left\{ \underbrace{\left\{ \frac{C_i + C_3 + C_{13}}{l_i} \mid i \in \{4, \dots, 11\} \right\}}_{\text{predefined VLCs}} \cup \underbrace{\left\{ \frac{C_5 + C_3 + C_{12}}{l_{12}} \right\}}_{\text{escape sequence}} \right\} \\ &= \min \left\{ \frac{C_4 + C_3 + C_{13}}{3}, \frac{C_5 + C_3 + C_{13}}{6}, \frac{C_6 + C_3 + C_{13}}{7}, \right. \\ & \quad \frac{C_7 + C_3 + C_{13}}{8}, \frac{C_8 + C_3 + C_{13}}{9}, \frac{C_9 + C_3 + C_{13}}{10}, \\ & \quad \left. \frac{C_{10} + C_3 + C_{13}}{11}, \frac{C_{11} + C_3 + C_{13}}{17}, \frac{C_5 + C_3 + C_{12}}{24} \right\} \\ &= \frac{C_{v'} + C_3 + C_{k'}}{l_{w'}} \end{aligned} \quad (3)$$

with $k' = 12$ (if $l_{w'} = 24$) or $k' = 13$ (if $l_{w'} \neq 24$). The maximum space required by DC coefficients equals $24 \cdot n_{IBlocks}$. Equivalent to Equation 2, the best-case execution time of phase one (C'_{Phase1}) can be computed:

$$\begin{aligned} C'_{Phase1}(l) &:= n_{IBlocks} \cdot (C_1 + C_2) \\ &+ \left\lfloor \frac{l - \min\{24 \cdot n_{IBlocks}, l\}}{l_{w'}} \right\rfloor \cdot (C_{v'} + C_3 + C_{k'}) \end{aligned}$$

It is possible that only DC coefficients are present in an I picture. This is reflected by the expression $l - \min\{24 \cdot n_{IBlocks}, l\}$, which guarantees that the result of C'_{Phase1} is always greater than zero.

The difference between C_{Phase1} and C'_{Phase1} is the maximum overestimation O_{max} :

$$\begin{aligned} O_{max}(l) &:= C_{Phase1}(l) - C'_{Phase1}(l) \\ &= n_{IBlocks} \cdot (C_1 + C_2) + \left\lfloor \frac{l - 3 \cdot n_{IBlocks}}{l_w} \right\rfloor \cdot (C_v + C_3 + C_k) \\ &\quad - \left(n_{IBlocks} \cdot (C_1 + C_2) \right. \\ & \quad \left. + \left\lfloor \frac{l - \min\{24 \cdot n_{IBlocks}, l\}}{l_{w'}} \right\rfloor \cdot (C_{v'} + C_3 + C_{k'}) \right) \\ &= \left\lfloor \frac{l - 3 \cdot n_{IBlocks}}{l_w} \right\rfloor \cdot (C_v + C_3 + C_k) \\ &\quad - \left\lfloor \frac{l - \min\{24 \cdot n_{IBlocks}, l\}}{l_{w'}} \right\rfloor \cdot (C_{v'} + C_3 + C_{k'}) \end{aligned} \quad (5)$$

The result of this function (using actual figures) is the maximum overestimation in clock cycles. Because O_{max} is a linear function, increasing with the length of a picture, the overestimation is limited to a linear bound.

3.3.2 Phase one

During the first phase, the VLCs of the complete picture are being identified and decoded:

1. headers (slice and macroblock)
2. motion vectors
3. DCT coefficients (blocks)

With each element of the picture being identified, the WCET required to decode this element in the second phase is estimated by adding corresponding function calls for each part of the decoding process (similar to Section 3.1). These functions simply add the decoding time of each identified element of the picture to a global counter. The result, i.e. the WCET estimation for the second decoding phase, can be provided to the scheduler at the end of the first phase (see Fig. 2). The decoded elements of the picture are stored in a buffer in order to be reused during the second phase. Since each element being decoded during phase two is identified during the first phase, the calculated WCET of phase two does not suffer from overestimations at all.

3.3.3 Phase two

During phase two, the actual picture has to be decoded. In this phase, each macroblock is completely decoded before advancing to the next. Since the VLC decoding was performed during phase one, the remaining picture decoding includes only the dequantization, the inverse DCT, and motion compensation (in case of P or B pictures). The elements are read from the buffer.

4. Experimental evaluation

Since the VoD solution provides the exact execution times, the evaluation is focused on the two-phase approach for the live scenario, where overestimations of the execution time compared to the worst possible case could be significantly decreased. We tested the two-phase decoder with several MPEG-2 streams, and compared the results to a naive approach that simply extracts the theoretically possible worst-case (thus ignores all input data). Note, that taking the naive approach is identical to using fixed bandwidth allocation. The results shown in Fig. 4 represent the overestimation together with the overhead required to decode each stream compared the results using a naive approach (giving the percentage of the actually necessary amount). The average overestimation plus the overhead for each of the sample streams was below 17%, resulting in a resource utilization of more than 85% compared to more than 160% (less than 39% resource utilization) using the naive approach (see Fig. 5). These figures give the potential a real-time scheduler can gain compared to a fixed-bandwidth reservation.

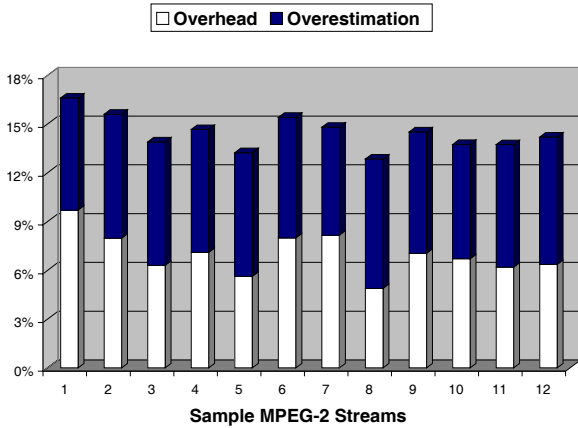


Figure 4. Overhead and overestimation of the two-phase decoder (percentage of the actual results)

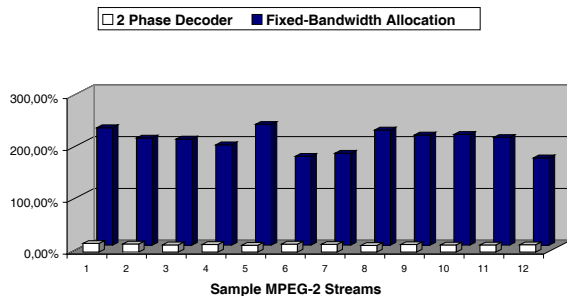


Figure 5. Results of the two-phase decoder compared to fixed-bandwidth allocation

5. Conclusion and future work

For the first time, the problem of worst-case execution times analysis of MPEG-decoding was addressed by this document, proposing solutions for different scenarios. Although not completely implemented for any possible MPEG-2 format, the two-phase decoder can be generalized to cover all possible MPEG-2 profiles and levels.

In the VoD scenario, our solution provides exact information about the execution time. For the live scenario, our two-phase approach significantly reduces the gap between the predicted worst-case and the actual execution times. The two-phase decoder represents a trade-off between reducing additional overhead and tightly estimating the WCET by decoding each picture during two phases. Before each phase, its WCET, calculated by the preprocessing function and the first phase respectively, is delivered to the scheduler. The overestimation of the WCET of the first phase is limited to a linear bound. The WCET of the second phase does not suffer from overestimations at all, and the computing

overhead, caused by the restructured decoding process compared to the original implementation, is kept low. Moreover, the two-phase decoder does not rely on specially encoded streams making it a flexible tool in any environment.

Currently, a real-time scheduler using the results presented in this document is being implemented. Its particular challenge is to cope with situations in which there is little time to react to significant changes in the resource allocation. Future work will also focus on a new version of the decoder, that can buffer more than one picture in order to provide information about longer video sequences. Other applications of the results presented in this document include encoders able to add WCET related information into the user-data fields during the encoding process and a program that can determine whether a given CPU is capable of decoding MPEG-2 streams.

References

- [1] N. Ahmed, T. Ntrajan, and K. R. Rao. *Discrete Cosine Transform*. *IEEE Trans. on Computers*, Vol. C-23:90–93, December 1984.
- [2] P. Altenbernd. *On the False Path Problem in Hard Real-Time Programs*. In *Proceedings of the 8th Euromicro Workshop on Real-time Systems*, pages 102–107, 1996.
- [3] N. Altman and N. Weideman. *Timing Variations in Dual Loop Benchmarks*. CMU/SEI-87-TR-22, Software Engineering Institute, Carnegie Mellon University, 1987.
- [4] A. Bavier, B. Montz, and L. Peterson. *Predicting MPEG Execution Times*. In *SIGMETRICS '98/PERFORMANCE '98. Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 131–140, 1998.
- [5] S. Bharrat and K. Jeffay. *Predicting Worst Case Execution Times on a Pipelined RISC Processor*. Technical Report TR94-072, Department of Computer Science, University of North Carolina - Chapel Hill, 1995.
- [6] R. Chapman. *Static Timing Analysis and Program Proof*. Dissertation, Computer Science Department, University of York, 1995.
- [7] P. Gopinath and R. Gupta. *Applying Compiler Techniques to Scheduling in Real-Time Systems*. In *IEEE Real-Time Symposium*, pages 247–256, 1990.
- [8] J. Gustafsson and A. Ermedahl. *Deriving Annotations for Tight Calculation of Execution Time*. In *Proceedings of the Euro-Par'97 Conference*, 1997.
- [9] M. G. Harmon, T. P. Baker, and D. B. Whalley. *A Retargetable Technique for Predicting Execution Time of Code Segments*. *Journal of Real-Time Systems*, 7(2):159–182, 1994.
- [10] ISO/IEC. *13818-2: Information technology - Generic coding of moving pictures and associated audio information - Part 2: Video*, 1996.
- [11] ISO/IEC. *N2932: MPEG-4 Video VM 14.0*, 1998.
- [12] I. I. Motorola. *PowerPC 604 RISC Microprocessor User's Manual*, 1994.

- [13] P. Puschner and C. Koza. *Calculating the Maximum Execution Time of Real-Time Programs*. *Journal of Real-Time Systems*, 1(2):159–176, September 1989.
- [14] M. Sjödin. *Response-Time Analysis for ATM Networks*. Licentiate Thesis, Department of Computer Systems, Uppsala University, 1995.
- [15] F. Stappert. *Predicting Pipelining and Caching Behaviour of Hard Real-Time Programs*. In *Proceedings of the 9th Euromicro Workshop on Real-time Systems*, 1997.
- [16] F. Stappert and P. Altenbernd. *Complete Worst-Case Execution Time Analysis of Straight-Line Hard Real-Time Programs*. *Journal of Systems Architecture*, 46:339–355, 2000.