Compression for Reduction of Off-chip Video Bandwidth

E.G.T. Jaspers\textsuperscript{a} and P.H.N. de With\textsuperscript{b}

\textsuperscript{a}Philips Research Laboratories, Eindhoven, The Netherlands
\textsuperscript{b}CMG Eindhoven / University of Technology, Eindhoven, The Netherlands

ABSTRACT
The architecture for block-based video applications (e.g. MPEG/JPEG coding, graphics rendering) is usually based on a processor engine, connected to an external background SDRAM memory where reference images and data are stored. In this paper, we reduce the required memory bandwidth for MPEG coding up to 67% by identifying the optimal block configuration and applying embedded data compression up to a factor four. It is shown that independent compression of fixed-sized data blocks with a fixed compression ratio can decrease the memory bandwidth for a limited set of compression factors only. To achieve this result, we exploit the statistical properties of the burst-oriented data exchange to memory. It has been found that embedded compression is particularly attractive for bandwidth reduction when a compression ratio 2 or 4 is chosen. This moderate compression factor can be obtained with a low-cost compression scheme such as DPCM with a small acceptable loss of quality.

Keywords: multimedia systems, embedded compression, transfer overhead, adaptive DPCM, data burst, bandwidth reduction, MPEG, memory communication

\textbf{Figure 1:} A video system with embedded compression.

1. INTRODUCTION
The processing architecture for block-based video processing functions such as compression standards (MPEG/JPEG), 3-D graphics rendering and field-rate conversion, is usually based on a processor engine consisting of various processors, which is connected to an external background SDRAM memory for the storage of images. This system represents a flexible multimedia platform with a shared memory, that enables simultaneous execution of several high-throughput stream-oriented video processing tasks (see Fig. 1). Due to the fast increase of required computational power of consumer systems, the data communication to and from the off-chip memory has become the bottleneck in the overall system performance (memory wall problem).

In recently developed systems, the memory communication and bandwidth problem was combated by communicating to several memory devices in parallel. However, this leads to the use of multiple memory controllers, more data connections requiring expensive chip packages, and therefore increased system costs. In this paper, we propose the combination of two techniques for substantially reducing the required memory bandwidth, thereby

\textsuperscript{a} egbert.jaspers@philips.com, \textsuperscript{b}peter.de.with@cmg.nl
keeping system costs low. First, we apply the newly developed technique for optimal mapping of the video into the memory by analyzing the actual memory accesses [1][2]. Second, we exploit an embedded compression technique in combination with our optimized mapping with the aim to further reduce the memory bandwidth.

Let us briefly consider why memory bandwidth is so scarce in video coding and processing systems. Most bandwidth-consuming tasks in multimedia platforms such as high-level MPEG coding and 3-D graphics, can be classified as video processing tasks with block-based pixel access at more or less random positions in the (off-chip) memory. These tasks, requiring large amounts of pixel data and intensive memory access, result in a large bandwidth overhead, due to the burst-oriented data transfer for currently available SDRAM devices. Particularly when the burst size approaches or exceeds the size of the required data block, such as e.g. a macroblock (MB) for MPEG coding, the overhead can be significant and double or even triple the required bandwidth. This paper shows the feasibility of an embedded compression technique to reduce the communication bandwidth and combines this with an optimal mapping of pixels into data bursts.

Although many compression techniques have been introduced, their usage for reducing the data bandwidth of the memory is not trivial. For example, a problem in computer systems is that the throughput of an application can degrade significantly when the working space of that application does not fit in the main memory. This results in an increased number of memory page faults so that the background memory on hard disk is accessed more often. As a possible solution, Roy et al. [3] implemented a concept of compressed memory pages to reduce the amount of disk accesses for the computer system. Although this may be true, the compressed memory pages need to be decompressed and written back into the memory when they are requested by the application. This process consumes extra memory access, so that the potential bandwidth reduction is influenced in a negative way. Moreover, it will be shown in the next section that a large grain size of compressed data packets such as the proposed memory pages, does not result in bandwidth reduction. Another application for embedded compression is presented in [4]. This approach applies lossless compression to relatively large data packets of 1 Kbyte in a high-complexity memory-controller chip to increase the memory capacity and is mainly intended for general-purpose computer servers. This functionality is located between the main memory and a large cache memory. Because the data traffic between the cache and the main memory is usually based on relatively large-grain packets. For the same reasoning as in the previous example, the compression gives a penalty in bandwidth, but because the large packet size there may be sufficient net gain. In this paper, where we concentrate on stream-based media processing, caching techniques offer only limited improvement because streaming data is commonly used only once within a limited time interval. Our objective is to develop an inexpensive solution without the need for substantial cache memory.

The use of compression for embedded memory applications has been studied primarily for reducing the required memory space. The usage of such techniques for memory bandwidth reduction, is not straightforward. For example, in [5], segments of nine macroblocks (MBs) were compressed into a fixed-sized block. Consequently, additional memory accesses are necessary to address individual MBs. In [6], a simple embedded DPCM technique is applied in HDTV decoding for memory capacity compression. Although this paper presents the measurement of bandwidth to derive the utilization memory bus, it does not further analyze the reduction in bandwidth. Van der Schaar et al. [7] proposed similar low-cost compression algorithms without bandwidth optimization and

![Figure 2: Memory access of a macroblock including the transfer overhead.](image-url)
showed later [8] that the obtained compression factor only partially aids in bandwidth reduction. In our paper, we quantify results on the feasibility of low-cost compression schemes (e.g. [9]) for reduction of the memory communication. Furthermore, we propose an optimization technique to find all feasible compression factors that reduce the bandwidth between the memory and the video processing. This reduction which can be as high as 67% for a compression ratio of four, can be exploited to enhance the system quality, reduce costs and/or add extra functionality.

Recent SDRAM devices use relatively large data bursts (e.g. 8 words) to improve the effective bandwidth at the expense of overhead, particularly when the aforementioned burst size approaches or exceeds the size of the required data block, e.g. a MB for MPEG coding. The overhead is caused by fetching more data than strictly required, i.e. all underlying data bursts (see Fig. 2). An embedded compression technique can be used to store more video pixels into a single data burst. However, as already mentioned, the reduction in bandwidth is not proportional with the compression ratio. Since the compression technique increases the amount of pixels contained in a single data burst, an additional overhead is introduced which has to be compensated by the compression ratio. The results in this paper show that the bandwidth can be substantially reduced for newly developed media systems with a wider memory bus (≥ 32 bits). We have obtained a reduction of the memory bandwidth by 38% in an MPEG decoding system without sacrificing visual picture quality (limited compression ratio two).

The paper is divided as follows. Section 2 discusses the behaviour of SDRAM memory and explains how pixels can be mapped into the memory. Section 3 addresses the compression algorithm and proves the feasibility of embedded compression. Section 4 briefly outlines our new mapping optimization technique [1][2] which is extended for use with embedded compression. Section 5 explores all possible solutions. Finally, we will conclude with the results.

2. MAPPING OF PIXELS INTO THE MEMORY

The efficiency of the memory access depends on two principal elements: the burst-oriented communication and the memory segmentation into separate banks. With respect to the first element, the utilization can be optimized when video data is accessed at the grain size of data bursts (e.g. a 64-bit bus and a burst length of eight words, leading to 64 bytes). These data bursts represent non-overlapping memory blocks that contain pixel data and which can only be accessed as an entity, referred to as data units. Consequently, small data-block accesses result in a large amount of pixel overhead, because complete data units need to be fetched from memory. Moreover, the amount of pixel overhead also depends on the aspect ratio of the data units. For example if a MB of 16 × 16 samples is requested, data units of 64 × 1 result in more overhead than data units of 8 × 8, although the size is equal. In [10], a technique is proposed to determine the optimal aspect ratio of the data units by analyzing the application software model only, without considering data dependencies. In our initial work [1][2] an optimal mapping of the video into the memory is determined by analyzing the actual memory accesses, so that data dependencies are taken into account. Note that for our objective, the size S of a data unit does not

![Figure 3: Mapping of video onto memory data units considering both progressive and interlaced accesses.](image-url)
coincide with the size $B$ of a data burst. For example, data units of $32 \times 4$ pixels can be compressed into a 64-byte data burst.

The second principal element for bandwidth efficiency is the organization into memory banks, commonly applied in all modern memory devices. Efficient communication can only be achieved when the memory banks are accessed alternately. Consequently, adjacent video pixels are interleaved in the memory banks. Furthermore, the successive odd and even lines of an interlaced video signal are mapped in different banks of the memory to prevent additional overhead when both progressive and interlaced video data blocks are requested by the video processing. The resulting mapping strategy evaluated for an MPEG-2 decoder, using $16 \times 4$ data units, is shown in Fig. 3. Note that an interlaced field requires interleaved access to all banks. Hence, alternating addressing of the memory banks for both progressive and interlaced video is achieved. A more detailed overview of pixel mapping into banks is given in [1][2].

3. EMBEDDED COMPRESSION

The previous section discussed how burst-oriented memory communication influences the optimal mapping of pixels onto the address space. This section will show that such a constraint is exploited to make embedded compression feasible for reduction of memory bandwidth.

Since compression techniques exploit spatial correlation between pixels, it is usually applied for a group of pixels. For example, when nine MBs are encoded as one entity [3], the entity has to be partially decoded before the value of a certain pixel can be determined. For retrieval of an arbitrary MB in the picture, several compressed data entities may have to be decoded because they all contain part of the requested MB. Obviously, this does not help in the reduction of the memory bandwidth, particularly if random access to the memory is required.

Let us consider an example that shows why embedded compression for bandwidth reduction is not easily obtained. We have measured the memory bandwidth of an MPEG-2 decoder.

If groups of $16 \times 8$ pixels are clustered as an access entity, 113% more data is transferred than was strictly required for the decoding process. Consequently, a compression ratio of 2.13 is necessary to accomplish the break-even point and even more for a net bandwidth reduction. However, when using an SDRAM-based memory, an overhead is already present due to burst-oriented storage. Table 1 indicates this overhead, dependent on the size of the communicated data bursts. For these numbers, the aspect ratio of the data-units are optimized for minimum transfer bandwidth. The bandwidth percentages show the transfer bandwidth relative to the bandwidth that is strictly required for the decoding. In the sequel of this paper we call these numbers the relative transfer bandwidth (RTB). Since the data-burst size is given by the architecture, compression becomes

<table>
<thead>
<tr>
<th>size [bytes]</th>
<th>optimal dimension</th>
<th>requested data [%]</th>
<th>transferred data 6-100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>(16 x 1)</td>
<td>100</td>
<td>131</td>
</tr>
<tr>
<td>32</td>
<td>(8 x 4)</td>
<td>100</td>
<td>149</td>
</tr>
<tr>
<td>64</td>
<td>(16 x 4)</td>
<td>100</td>
<td>172</td>
</tr>
<tr>
<td>128</td>
<td>(16 x 8)</td>
<td>100</td>
<td>213</td>
</tr>
</tbody>
</table>

more feasible: only the additional overhead due to compression needs to be compensated. For example, if data units of $16 \times 8$ are compressed into 64-byte data bursts, the RTB increases from 172% to 213%. Therefore, the break-even compression ratio changes to $2.13/1.72 = 1.24$, which is feasible.

Up to this point, we discussed the properties of an SDRAM memory that result in block-based (bursts) storage of data and therefore increase the bandwidth requirement significantly. Furthermore, we have explained how
this block-based storage can be exploited to reduce the bandwidth requirement by means of compression. However, the suitability of compression schemes is bound by the constraints we have developed so far. Firstly, we found that the burst size determines the size of the compressed data entity. Secondly, the data must be easily accessible at regular address positions, thereby leading to a fixed compression ratio. Hence, this leads to fixed-sized input data units (data-units size $S$) and output blocks (data-burst size $B$). Let us now discuss a compression algorithm that satisfies the aforementioned constraints, presented by Bayazit et al. [9]. This technique is based on an adaptive DPCM algorithm, which can achieve a 50% memory compression ratio with no distinguishable loss of visual quality. The paper describes the independent compression of one data block consisting of one luminance block of $16 \times 2$ samples and two chrominance blocks of $8 \times 1$ samples, thus 48 bytes in total. Because our system assumes separate storage of the luminance and chrominance data, it is required to independently compress luminance and chrominance components. This requirement does not limit the suitability of the algorithm and no performance degradation is expected for this reason. Another difference for the applicability of the proposed algorithm is the size of the data units. Data units of 48 bytes are rather small for obtaining sufficient compression. Section 5 will show that most suitable data-unit sizes for 32-byte data bursts ($B = 32$) are $S = 64$ bytes (and larger). For larger data bursts the most suitable size is even larger. Consequently, the algorithm will be able to exploit more correlation when applied for our purpose.

The experiments as described in the above-mentioned paper show that the compression scheme was not matched to the size of the memory bursts. The 48-byte data blocks were compressed with a factor 1, 1.33, 1.6, and 2, resulting in compressed data entities of 48, 36, 30, and 24 bytes, respectively. Because the sizes of these data entities do not match with the alignment grid of the data bursts, these compression ratios lead to sub-optimal solutions; i.e., more data bursts are transferred to access a compressed data entity. The results of the experiments show a bandwidth reduction of 12.5% for a compression ratio of two. The corresponding results on picture quality show a degradation of 0.97 dB for a MP@ML MPEG-2 video decoder, when decoding a high-quality 9-Mbps bitstream. For a bitstream that was coded at 4 Mbps, the quality degraded with only 0.34 dB. Subjective tests revealed high-quality video with imperceptible artifacts. For our system we target to decorrelate larger data blocks and hence expect an even better picture quality. Moreover, the results in the next section will show a significantly higher bandwidth reduction than in [9].

4. BANDWIDTH CALCULATIONS

In our experiments we used the implementation of an MPEG-2 decoder to statistically analyze the behaviour of the data communication. In this section we elaborate on the communication dependencies for computing the overhead. To derive the memory bandwidth, the calculation model as proposed in [1][2] is used. The calculated result from this model represents the transfer bandwidth relative to the bandwidth that is strictly required for the decoding (RTB). However, in this model, embedded compression is not considered. Consequently, the size $S$ of a data unit is assumed to be equal to the data-burst size $B$ ($S = B$). In the following, we extend the above-mentioned model for the use of embedded compression. For this purpose, we introduce the compression ratio $c_s$, where $S$ stands for the data-unit size. The value of the compression equals the ratio between the data-unit size and the data-burst size, thus:

$$c_s = \frac{S}{B} \quad (1)$$

Below, we list the parameters on which the calculations for the RTB depend, including the compression ratio $c_s$:

- the dimensions of the requested data blocks, $B_x \times B_y$;
- the dimensions of the data units, $(M, N)$;
- the interface factor of the requested data blocks;
- the probability of their occurrence, $P(B_x \times B_y)$;
- the probability distribution of their positions, $P_{B_x \times B_y}(m, n)$;
- the compression ratio, $c_s$. 
For the bandwidth calculation, the set of possible data-block requests $V$ has to be divided into a subset of progressive data block requests $V_p$ and a subset of interlaced data block requests $V_i$, such that $V = V_i \cup V_p$. This separation is necessary because the calculations for both contributions are slightly different. We denote the average RTB of progressive data-block requests by $\tilde{\sigma}_p(M, N, V_p)$ and for interlaced data-block requests by $\tilde{\sigma}_i(M, N, V_i)$. Both contributions are already probability weighted, so that the total average RTB is:

$$\tilde{\sigma}(M, N, V) = \tilde{\sigma}_i(M, N, V_i) + \tilde{\sigma}_p(M, N, V_p).$$

(2)

To achieve the minimal bandwidth requirements for a given application, the dimensions of the data units $(M, N)$ are optimized.

In this paper we consider an MPEG-2 decoder as an example for our experiments. For this application the set of data blocks is:

$$V_p = \{(16 \times 16), (17 \times 16), (16 \times 17), (17 \times 17), (16 \times 8), (18 \times 8), (16 \times 9)\}$$

$$V_i = \{(16 \times 16), (17 \times 16), (16 \times 17), (17 \times 17), (16 \times 8), (18 \times 8), (16 \times 9), (18 \times 9), (17 \times 8), (17 \times 9), (16 \times 4), (18 \times 4), (16 \times 5), (18 \times 5), (M \times N)\}$$

The large variety of data-blocks requests is caused by the MPEG standard and depends on field/frame prediction, luminance/chrominance data and the sub-pixel accuracy. In our experiments, we consider the reading of prediction data for motion compensation, the writing of the motion-compensated result, $(16 \times 16) \in V_p$, and the reading for interlaced display of the video, $(M \times N) \in V_i$. For the last aspect, it is assumed that the display unit contains line memories to read the block-based data units of $(M \times N) \in V_i$ and to display the video lines sequentially. To acquire representative results for an average MPEG-2 bit stream, a large set of test bit streams, containing in total 732 frames, is used to derive the optimum.

To determine $P(B_x \times B_y)$, the amount of occurrences of each type of data-block, all requests are measured at the memory interface, thereby feeding only the data dependencies into the model.

The probability distribution of the positions of the data blocks $P_{B_x \times B_y}(m, n)$, is defined as the probability that the upper-left corner pixel of a requested data block $B_x \times B_y$ is positioned at any location $(x, y)$, satisfying the condition: $(x \text{ mod } M = m) \text{ AND } (y \text{ mod } N = n)$. Hence, a low-complexity bookkeeping of the occurrences at position $(x \text{ mod } M, y \text{ mod } N)$ is used to determine $P_{B_x \times B_y}(m, n)$. This probability distribution highly depends on the dimensions of the data units. Large differences may occur in the result, due to the $16 \times 16$ MB grid for MPEG and the high probability of the zero-motion vectors. For example, if $(M, N) = (32, 2)$, the probability that $(x \text{ mod } 32 = 0) \text{ AND } (y \text{ mod } 2 = 0)$ is relatively high. However, for data-unit dimensions that are not aligned with the MB grid, e.g. $(M, N) = (12, 5)$, the probability distribution of the positions of the data-blocks is totally different (see both examples in Fig. 4).

![Figure 4: 32 × 2 (a) and 12 × 5 data units (b) overlaid on a MB grid.](image-url)
At this stage, we have discussed the parameters that are necessary for the calculations and can derive the RTB. For the set of interlaced data-block requests, the following equation applies:

$$\bar{\sigma}_i(M, N, V_i) = \frac{\sum_{B_x \times B_y \in V_i} P(B_x \times B_y) H(M, N, V_i)}{c_S \cdot \sum_{B_x \times B_y \in V_i} P(B_x \times B_y) \cdot B_x \cdot B_y},$$

(3)

with

$$H(M, N, V_i) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} P_{B_x \times B_y}(M, N) \cdot \left(1 + \left[\frac{B_x + m - 1}{M}\right]\right) \cdot \left(1 + \left[\frac{B_y + n - 1}{N}\right]\right).$$

The summation in numerator in Eq. (3) represents the amount of transferred pixels including the overhead, whereas the summation in the denominator represents the amount of pixels that is strictly required for the decoding without the overhead. $c_S$ indicates the compression ratio. Note that the calculation without $c_S$ resembles the amount of transferred pixels relative to the amount of pixels that are strictly required for the decoding. Since this is directly proportional to the bandwidth, the equation with $c_S$ points to the effective bandwidth relative to the bandwidth that is strictly required for decoding without compression. The RTB calculation for the set of progressive data-block requests is similar to Eq. (3) but $V_i$ becomes $V_p$ and $H(M, N, V)$ is defined according to:

$$H(M, N, V_p) = \sum_{m=0}^{M-128} \sum_{n=0}^{N-1} P_{B_x \times B_y}(M, N) \cdot \left(1 + \left[\frac{B_x + m - 1}{M}\right]\right) \cdot \left(2 + \left[\frac{B_y/2 + [n/2] - 1}{N}\right]\right) \cdot \left[\frac{B_y/2 + [n/2] - 1}{N}\right].$$

5. EXTRATION OF FEASIBLE SOLUTIONS

The results of the previous section present a calculation model for the memory communication, featuring the data dependencies and the usage of embedded compression. Although the feasibility of embedded compression for reduction of the memory bandwidth is already proven by the example in Section 3, this section uses the calculation model to identify all feasible solutions. The method to systematically determine these solutions comprises of the following steps:

- determination of the optimal data-unit configuration for zero compression;
- determination of the optimal data-unit dimensions for an incremental value of the compression ratio;
- pruning of the non-feasible data-unit configurations.

The outcomes of the first step is already presented in Table 1. For example, the optimal dimensions $(M_B, N_B)$ for 64-byte data bursts $(B = 64)$ and a data-unit size $S = B$, are formalized as follows:

$$(M_B, N_B) = \{ (m,n) : m \in [1..B], n \in [1..B] : \downarrow \sigma(m, n, V) \land m \cdot n = B \},$$

(4)

where $\downarrow$ denotes the minimum. For example, the minimum RTB for data units containing 64 pixels equals $\sigma(M_B, N_B, V) = 172\%$.

In the second step, the optimal data-unit dimensions are determined for increasing compression ratio. Note that the compression ratio $c_S$ is given by Eq. 1. Thus, for a fixed data-burst size $(B = 64)$ the data-unit size is incrementally increased from the size of the data burst up to four times the burst size. Note that this resembles an incremental increase of the compression factor. For each $S \in (B..4B]$ the optimal data-unit dimensions can be determined by:

$$(M_S, N_S) = \{ (m,n) : m \in [1..S], n \in [1..S] : \downarrow \sigma(m, n, V) \land m \cdot n = S \}.$$  

(5)
As a result from this calculation the RTB value for the optimal dimensions \((M_S, N_S)\) for each \(S\) is \(\bar{\sigma}_5(M_S, N_S, V)\). Fig. 5 shows these values as function of the data-unit size. Because most solutions do not result in less data transfer than in the case without compression, they have been removed for an improved visualization of the results. Consequently, all shown solutions satisfy the following equation:

\[
\bar{\sigma}_5(M_S, N_S, V) < \bar{\sigma}_5(M_B, N_B, V) \quad \text{with} \quad S \in (B.AB].
\]

Figure 5: Minimal RTB as function of the data-unit size \(S\) and a burst size of \(B = 64\) bytes.

In the third and final step, pruning of the found solutions can be applied. This pruning means in Fig. 5 that, starting at the left side and going to the right, only those solutions are adopted that give a lower RTB than the previously selected point. This means that all solutions with a larger compression ratio than other solutions while having less bandwidth-reduction gain, are removed. Since the picture quality as function of the compression ratio is generally a monotonic decreasing function, it can be concluded that the removed solutions have a lower picture quality than the remaining solutions while consuming more bandwidth. After the pruning process all remaining solutions satisfy the following condition:

\[
\left( \forall \sigma_1(M_{S_1}, N_{S_1}, V), \sigma_2(M_{S_2}, N_{S_2}, V) : S_1 \in (B.AB], S_2 \in (S_1..AB] : \bar{\sigma}_5(M_{S_1}, N_{S_1}, V) > \bar{\sigma}_5(M_{S_2}, N_{S_2}, V) \right).
\]

The remaining data-unit configurations represent feasible solutions and are shown in Fig. 6. The label at each point indicates the optimal data-unit dimension giving the result. This picture enables a tradeoff between compression ratio and bandwidth reduction. Note that data units of 16 \(\times\) 8, reduce the relative transfer bandwidth from 172\% to 106\%. This compression ratio provides the largest amount of bandwidth reduction per unity of compression. Moreover, the data-unit dimensions are powers of two, which makes it even more attractive from an implementation point of view.

Figure 6: Feasible data-unit configurations for compression into 64-byte data bursts.
To explore the complete design space, the above-described method is also applied to an MPEG-2 decoder system with a external memory having 32-byte and 16-byte data bursts; i.e. $B = 32$ and $B = 16$, respectively. The outcomes are depicted in Fig. 7 and Fig. 8. Notice that the graphs in Fig. 6, 7 and 8 look very similar. For example, in all figures, the maximum amount of bandwidth reduction per compression unit is positioned at compression ratio two. For this compression ratio, the optimal data-unit dimensions are aligned with the MB grid in both horizontal and vertical direction, thereby enabling memory requests with low overhead at relatively high probability. Moreover, also the size of the data unit is an important parameter for feasibility. A data unit of $16 \times 8$ can easily be compressed with a factor two, while maintaining a high picture quality. For data units of $8 \times 4$ this is less straightforward.

6. RESULTS AND CONCLUSIONS

Many current multimedia systems intended for applications such as e.g. MPEG coding and 3-D graphics rendering, feature double-data-rate (DDR) SDRAM with bus widths up to 64 bits. These expensive memory configurations are adopted to obtain sufficient communication bandwidth, which is demanded by the continuous increase of computational power. This paper shows the feasibility of embedded compression for reducing the previously mentioned costly bandwidth bottleneck.

Our experiments with an MPEG-2 decoder show that the amount of data transferred to and from the memory is 172% of the data that is strictly required for the decoding, using the optimal mapping of groups of pixels (data units) to be compressed into data bursts. However, in most currently used systems a straightforward linear mapping of $64 \times 1$ pixels into data bursts is applied, resulting in a relative transfer bandwidth of even 341% [1][2]. Due to the trend of increasing bandwidth requirements, the memory data bus is becoming wider. Consequently, the size of the data bursts grows, leading to even more transfer overhead. Fortunately, larger
blocks can be decorrelated more efficiently, which makes the use of embedded compression for memory bandwidth reduction increasingly attractive.

Unfortunately, it has been found that compression not necessarily leads to reduction of the memory bandwidth. Due to the block-based storage of compressed data entities, the bandwidth may even be increased instead of reduced. This phenomenon limits the amount of feasible solutions for using embedded compression. Moreover, it has also been found that certain data-unit sizes offer less bandwidth reduction than other solutions, while giving less picture quality. Obviously, these solutions are therefore not attractive and have been omitted for feasibility. This has resulted in a limited amount of feasible compression ratios corresponding with a limited amount of optimal data-unit dimensions.

For our consumer application, it is required to provide a low-cost compression scheme to reduce bandwidth without sacrificing visual picture quality. For a compression factor four, a bandwidth reduction is established of 53%, 64% and 67% for 64-byte, 32-byte and 16-byte data bursts, respectively. However, for all three burst sizes, a compression ratio of two gives the largest amount of bandwidth reduction per unit of compression. For this compression factor an algorithm from Bayatla et al. [9] that is based on adaptive DPCM has been proven to be feasible. We propose to slightly modify the algorithm by matching the dimensions of the independent data units to the properties of the burst-oriented SDRAM memory, thereby enhancing the bandwidth reduction from 12.5% to approximately 40%. More precisely, for a compression ratio of two, the bandwidth is reduced with 38%, 42% and 43% for 64-byte, 32-byte and 16-byte data bursts, respectively. Summarizing, our data-dependent communication model decreases the memory bandwidth of the MPEG decoder from 341% to 172% and the embedded compression accounts for a further reduction to 106%, thereby gaining a total bandwidth reduction of 69%.

With respect to picture quality, the following conclusion applies. For data-burst sizes of 64 and 32 bytes, we compress 128 and 64 bytes, instead of the 48 bytes as presented in [9]. Therefore, we can expect the picture-quality degradation in the MPEG decoder caused by the embedded compression to be less than 0.34 dB as indicated in the same paper.

A side benefit of using an embedded compression scheme with a fixed compression ratio is that it reduces the required memory size proportionally to the compression ratio. Thus where usually three picture memories are required for MPEG decoding, only half of it is necessary when applying embedded compression with a factor two.

The proposed techniques for reducing the bandwidth bottleneck of external memory can be applied to a broad class of block-based video processing applications. When adopting embedded compression for decoding of high-definition video pictures, the bandwidth reduction is as high as 165 or 275 MB/s for a compression ratio of two or four, respectively. Since this substantial improvement in memory bandwidth is guaranteed, it can be exploited easily for reducing the system cost, to improve the picture quality, or to extend the functionality.

REFERENCES