Journal of Signal Processing Systems (2022) 94:693–708
https://doi.org/10.1007/s11265-021-01727-2

Architecture of a Low Latency H.264/AVC Video Codec for Robust ML based Image Classification

How Region of Interests can Minimize the Impact of Coding Artifacts

Fritjof Steinert[1,2] · Benno Stabernack[1,2]

Received: 16 April 2021 / Revised: 24 September 2021 / Accepted: 30 November 2021 / Published online: 31 January 2022

Abstract
The use of neural networks is considered as the state of the art in the field of image classification. A large number of different networks are available for this purpose, which, appropriately trained, permit a high level of classification accuracy.Typically, these networks are applied to uncompressed image data, since a corresponding training was also carried out using image data of similar high quality. However, if image data contains image errors, the classification accuracy deteriorates drastically. This applies in particular to coding artifacts which occur due to image and video compression. Typical application scenarios for video compression are narrowband transmission channels for which video coding is required but a subsequent classification is to be carried out on the receiver side. In this paper we present a special H.264/Advanced Video Codec (AVC) based video codec that allows certain regions of a picture to be coded with near constant picture quality in order to allow a reliable classification using neural networks, whereas the remaining image will be coded using constant bit rate. We have combined this feature with the ability to run with lowest latency properties, which is usually also required in remote control applications scenarios. The codec has been implemented as a fully hardwired High Definition video capable hardware architecture which is suitable for Field Programmable Gate Arrays.

Keywords H.264 · Advanced Video Codec (AVC) · Low Latency · Region of Interest · Machine Learning · Inference · FPGA · Hardware accelerator

1 Introduction

Our application scenario is focused on the detection of persons floating in water (shipwrecked) as a first step in a successful sea rescue operation. The system is shown in Fig. 1. For this purpose an unmanned aerial vehicle (UAV) with a full HD (High Definition) camera pointing downwards is used to record the water surface. The intended UAV has a long flight duration and can fly up to 100 km away from the ground station. The video data is transmitted to the ground station via a radio link with a very limited bandwidth, especially with increasing distance to the station. Due to the limited bandwidth a video codec must be used. At the ground station, an operator searches the video image for persons in the water and other objects. For this, the operator is assisted by complex machine learning (ML) methods for the recognition of persons.

The proposed processing pipeline consists of:

1. An HD camera which captures the scene.
2. A Region of Interest (ROI) generation based on the captured scene e.g. using simple ML methods.
3. Encoding the video with the recognized ROIs.
4. Transmission of the bitstream over a limited wireless air link to the ground station.
5. Decoding the video at the ground station.
6. Classification of objects in ROIs using more complex ML algorithms. Additional object detection over the entire image to possibly identify objects not detected in the UAV.

✉ Fritjof Steinert
fritjof.steinert@hhi-extern.fraunhofer.de

Benno Stabernack
benno.stabernack@hhi.fraunhofer.de

1 Fraunhofer Institute for Telecommunications - Heinrich Hertz Institute (HHI), Einsteinufer 37, 10587 Berlin, Germany

2 University of Potsdam, Potsdam, Germany

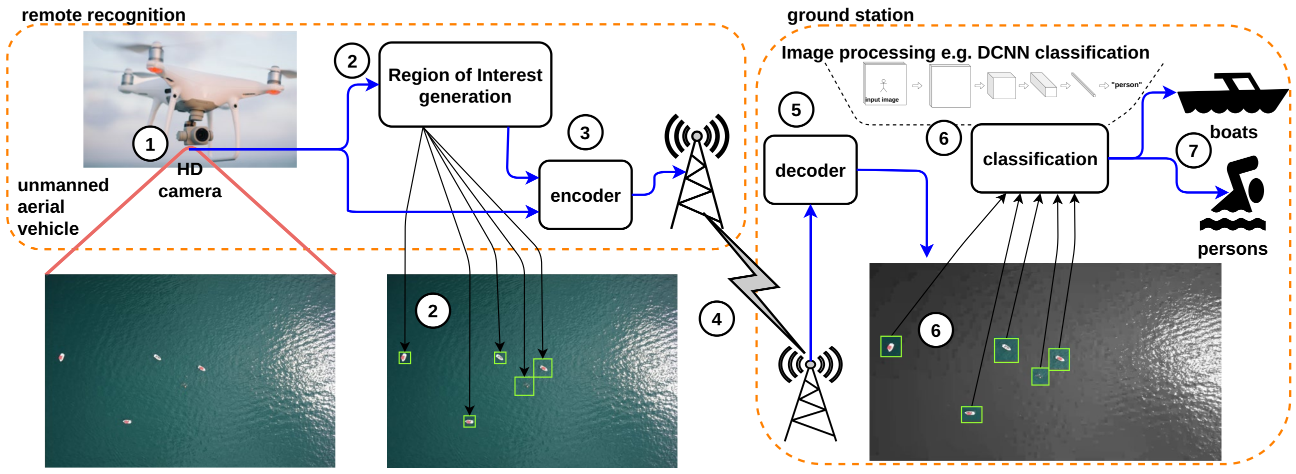

Figure 1 Sea rescue application scenario.

7. Display of the recognized object classes as a label for visual support of the operator.

Due to the limited computing power in the UAV, only simple ML methods can be used to detect the ROIs. A more detailed analysis with complex ML algorithms is performed in the ground station. Several requirements for the video codec are derived from this scenario:

R1: Real-time processing is necessary because the camera continuously streams video data.
R2: Since control data of the UAV is also transmitted via the radio link, rate control must be applied to ensure that the encoded video data does not exceed a defined bandwidth. At the same time, the available bandwidth should be used in the best possible way for a good video quality.
R3: Since ML requires a high image quality with few coding artifacts for a reliable detection rate, potentially interesting regions should be detected in the UAV and encoded with a better quality.
R4: The encoding has to be done with low latency so that ROIs set manually by the operator cause a timely change of the encoded video.
R5: The codec should also run on low-end FPGAs (Field Programmable Gate Array) and therefore be implemented with low complexity.
R6:In order to be able to manually detect shipwrecked persons not detected in the UAV due to the simple ML algorithm, the entire image and not only the ROIs should be transmitted.

Other technical applications of video coding in the automotive, railroad or marine sector have similar requirements. For example, in autonomous driving applications, ultra-low latency combined with high image quality in the regions of interest and constant data rate is relevant for ML based object recognition, e.g., of cyclists. At the same time, a large number of camera streams are to be transmitted via a single network for cost reasons, so that the available bandwidth is limited and encoding is mandatory.

In this paper, which is based on our publication [36], we present the hardware architecture of a low-complexity, low-latency video codec based on H.264/Advanced Video Coding (AVC). The codec can be implemented on low-cost FPGAs, yet still providing sufficient coding efficiency for the given application scenario and satisfying the stated requirements. We also show its integration into a distributed image processing system. Compared to our conference paper, we extended the explanation of the process pipeline as well as related work and included the external memory requirements for encoder and decoder. To the results we added a more extensive resource review, a comparison with related work, and analysis of power consumption. In addition, we presented the system integration of the codec into a hardware framework.

The structure of the paper is as follows: In Sect. 2 we present the influence of video coding quality on image processing algorithms. We also discuss the contributions of individual components of a signal processing pipeline to video coding delay and the importance of low latency. Section 3 describes the architecture of the low-latency, low-complexity video codec for FPGAs, separately for encoder and decoder. Section 4 provides our results and the related discussion of the FPGA implementation in terms of resource utilization, clock frequency and the influence of ROIs on image quality. In Sect. 5 we cover the integration of our codec into a distributed system via network interfaces. The remaining part of the paper is dedicated to a conclusion and our future work.

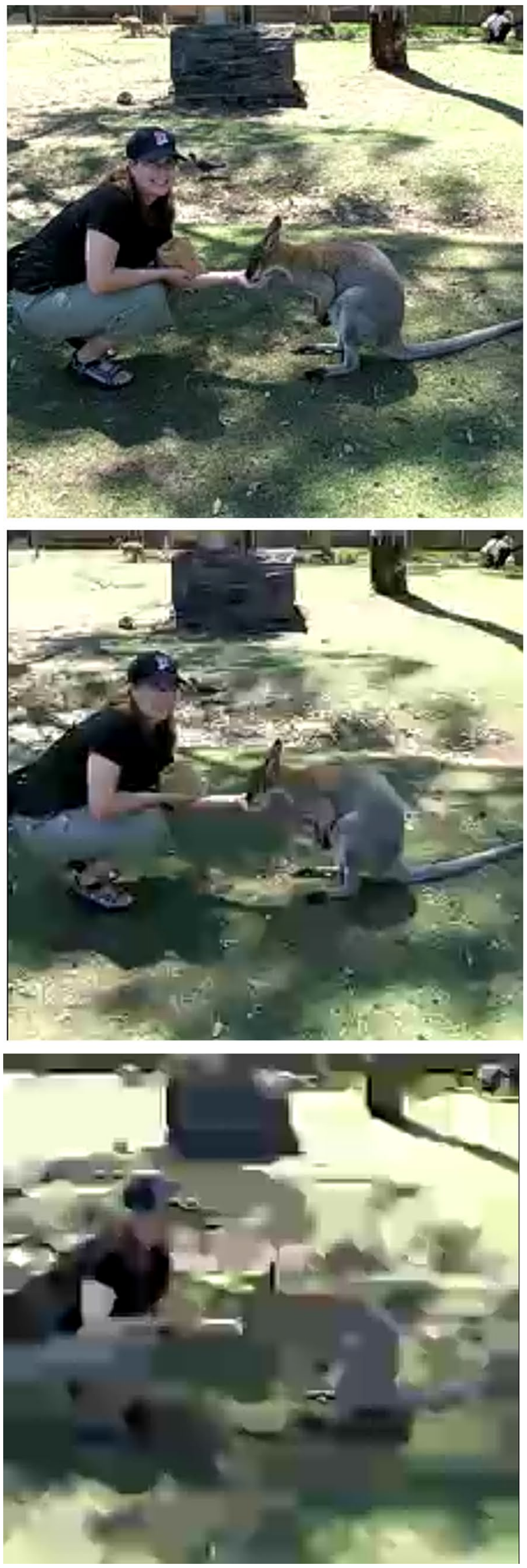
Figure 2 Example images from top to bottom with 33 dB PSNR Y, 26 dB PSNR Y and 22 dB PSNR Y.

2 Background and Related Work

2.1 Influence of Video Coding Quality on Image Processing Algorithms

A fundamental problem with video coding is the fact that the drastic reduction in the data rate is based on lossy coding methods, i.e. as the compression rate increases, the image quality deteriorates and leads to visible coding artifacts. Due to the applied coding methods this means, that achieving a constant bit rate (CBR) results in varying video quality and when aiming for a constant video quality the resulting bit rate cannot be held constant (variable bit rate, VBR). Typically, CBR is preferred for transmission channels with a specific channel capacity, while VBR is often used in cases where consistent video quality is required, e.g. Blu-ray or other storage media. Using video coding in classical applications scenarios like video streaming, the coding quality will be judged by the viewer who is in most cases willing to accept artifacts in favor of using low bit rate transmission channels. In contrast to these scenarios technical applications typically apply image processing algorithms on the decoded video streams. In this case the resulting video quality must be held constant to achieve the best results. An often used measure to indicate the quality of video coding is PSNR (peak signal to noise ratio), where a higher value means better video quality. With the advent of the broad usage of DCNN (deep convolutional neural network) as image processing algorithms for various applications, our following simple experiment shows that the higher the PSNR, the higher is the resulting confidence of the classification, which obviously deteriorates when using error-prone low bit rate channels.

Figure 2 shows three images that have been coded using H.264/AVC Baseline Profile with three different target PSNR values. In Table 1 the inference results of an imagenet-trained MobileNetV2 DCNN [32] using the three images depicted in Fig. 2 are given as global confidence for the predicted class. The example given in Table 1 is not sufficient to make a general statement on the influence of video coding artifacts, but it is evident that a drastic degradation of the confidence is occurring. It is in that sense obvious that by using high-quality images for the training of neural network parameters, video coding artifacts typically are not taken into account. A number of papers have been published in which the problem has been analyzed in depth.

The authors in [12] examine the relationship between the recognition accuracy of different neural networks and the image quality delivered by a corresponding camera. An

Table 1 MobileNetV2 inference results

Original			**33 dB PSNR Y**		**26 dB PSNR Y**		**22 dB PSNR Y**	
#	confidence	prediction	confid.	prediction	confid.	prediction	confid.	prediction
1	0.998468	104 (wallaby, brush kangaroo)	0.602415	104	0.078762	268 (Mexican hairless)	0.160643	847 (tank, ...)
2	0.000088	273 (dingo, ...)	0.034602	341 (hog, ...)	0.037112	341	0.064593	895 (warplane, ...)
3	0.000083	383 (Madagascar cat, ...)	0.027936	286 (cougar, ...)	0.030874	346 (water buffalo, ...)	0.033603	438 (beaker)
4	0.000082	373 (macaque)	0.023066	394 (sturgeon)	0.028397	150 (sea lion)	0.028306	403 (aircraft carrier, ...)
5	0.000063	105 (koala, ...)	0.016754	227 (kelpie)	0.026863	148 (killer whale, ...)	0.023474	660 (mobile home, ...)

Ethernet-based camera is used, which also applies H.264/AVC as the coding method. The application scenario in this case is the detection of vehicles in an automotive environment. An analysis of the recognition accuracy is carried out on the basis of different bit rates and different image sizes. Here too, the authors report that the recognition accuracy depends on the data rates. In this use case very high data rates were used, at which the coding artifacts are relatively moderate. Since the camera provides a constant bit rate a high probability exists in this scenario, that the image quality can be greatly reduced in certain cases, which means a non-deterministic behavior of the following object detection algorithms. Since the paper focuses foremost on the analysis, no solution to the problem is given. The authors in [7] discuss the problem related to applying a people detection DCNN to decoded video data. In this scenario the authors take multiple cameras with multiple streams into account, where the bandwidth needs to be shared between the different cameras. Based on object detection in one particular video stream the quantization parameter QP of the corresponding video encoder will be changed by a central server that is aware of the overall bandwidth, which means in turn that the data rate will be adapted to the needed image quality. The overall system has been simulated and shows good results regarding the object recognition. One drawback is the backchannel needed to adjust the rate control of the different video encoders accordingly. In [11] the authors discuss the influence of video compression artifacts on a FIR (far infrared) video stream for the purpose of pedestrian detection based on night vision technology. The authors analyzed MJPEG (Motion-JPEG) and H.264/AVC. AVC shows better performance at the same bit rates, which is not surprising since the standard provides a much higher coding efficiency compared to MJPEG by using more coding tools and temporal prediction methods. Here no solution is given besides the fact, that the authors vote for a certain data rate limit used by the video encoder as well.

The presented H.264 codec features ROIs to provide improved image quality in interesting image areas. At the same time, maximum bandwidth is guaranteed.

2.2 Video Coding Latency

Besides video coding quality in terms of PSNR, the most interesting non-functional video coding parameter for technical applications is the achieved overall coding latency, which describes the time needed to process a video signal from the input of the encoder to the corresponding video signal at the output of the video decoder, often referred to as the so-called glass-to-glass delay. This parameter is mostly important for applications where an interaction between the receiver and the transmitter side takes place. In these applications latencies much lower than one frame period (e.g. for 25Hz frame rate the time period for one frame is 40ms) are required, which could be in the range of only several milliseconds. In the past, this problem has been typically solved by the transmission of uncompressed digital video signals such as LVDS (Low Voltage Differential Signaling) which provides the lowest latency possible. The drawback of this approach is the lack of a seamless integration into existing digital network infrastructures such as Ethernet due to the much higher bandwidth demanded or the requirement for a wireless link, which is typically not capable of high data rates in the Gbps range. E.g. in the automotive domain there is a strong need to reduce cost and weight by reducing the number of copper wires in future cars to optimize fuel economy or battery life and overall costs. Service integration like Advanced Driver Assistance Systems (ADAS) [11] leads to the demand of transmitting video signals digitally over the same in-car bus system typically dedicated to control information. Due to high data rates of uncompressed video and the limited bandwidth of the bus systems used e.g. optical Ethernet, compression of the video signal is mandatory while still providing the same latency as the equivalent uncompressed or analog video links/systems. Other examples for applications with similar requirements are telemedicine [3], endoscopic medical imaging [5], online video gaming [40] or real-time steering of robotic equipment using wireless links for video transmission. The major burden when replacing digital uncompressed video links by video compression schemes is the desired low latency on one side and the video compression inherent accumulated coding delay caused by the different involved buffers on the other hand. As depicted in Fig. 3 these incorporated sources for delay are:

$$\Delta t_{all} = \Delta t_{VideoInput} + \Delta t_{Encode} + \Delta t_{OutputBuf} + \Delta t_{Network} + \Delta t_{InputBuf} + \Delta t_{Decode} + \Delta t_{VideoOutput} \quad (1)$$

where:

- Δt_{all} is the overall delay / latency.
- $\Delta t_{VideoInput}$ time to read in the video signal.
- Δt_{Encode} time to encode the captured video signal.
- $\Delta t_{OutputBuf}$ time the corresponding compressed data resides in the output buffer of the encoder.
- $\Delta t_{Network}$ time needed to transmit compressed data element over the network.
- $\Delta t_{InputBuf}$ time the corresponding compressed data resides in the input buffer of the decoder.
- Δt_{Decode} time to decode the received compressed data element.
- $\Delta t_{VideoOutput}$ time to send decompressed video signal to output device.

It is obvious, that besides the network latency, the major delay results from the codec and its corresponding buffers. Whereas the buffer size is mostly dependent on the mode (CBR or VBR) and the compression method used e.g. temporal predictive or intra predictive coding, the delay resulting from the codec is defined by its architecture. Typically a hardware architecture relies on a MB-based processing pipeline where the resulting latency can be significantly lower compared to the delay introduced from the accompanied bitstream buffers. The size of the bitstream buffer results from the requirements regarding the generation of a constant data rate at the output of the video encoder. Compression methods based on temporal predictive methods typically show a significant variation in the output data rate whereas intra predictive coding methods provide a more constant bit rate at the price of a overall higher data rate and lower coding efficiency. Regarding the encoding delay both methods intra and motion compensated temporal predictive modes can achieve the same coding delay, by carefully controlling the bitstream buffers. In [20] the authors describe a rate control scheme, which has been applied to an H.264/AVC intra only encoder, which provides very low latency by controlling the output buffer on macroblock (MB) level.

In our proposed codec architecture, a very low latency of less than 2 ms is achieved for encoder, decoder and the minimum required buffer sizes combined.

2.3 Hardware Architectures for Video Coding

One possible solution to provide the lowest achievable coding latency is to avoid compression methods relying on temporal prediction. In this case only intra frame predictive methods will be used by the encoder leading to high data rates and latencies which can be below one frame period. Besides compression standards that only provide intra type coding methods as JPEG-2000 or M-JPEG, most video coding standards can be used by applying only intra modes. Based on the results presented in previous sections, we decided to develop an H.264 codec that on the one hand provides good video quality and on the other hand meets the strict latency requirements. As mentioned above, the lowest possible latency will be achieved using intra-only codec approaches. In [6] an architecture is presented by the authors based on a quality scalable architecture featuring the possibility to adapt the video coding performance to the clock requirements of the underlying system. As a chip implementation (90 nm process) it can by used to encode D1, HD720 and HD1080 video sequences with 30 fps at clock frequencies of 23 MHz to 43 MHz, 71 MHz to 116 MHz and 152 MHz. Besides the high bandwidth inherent to intra-only codecs, it is characterized by the low latency required. The authors in [21] describe their architecture, which performs nearly with the same throughput numbers as our FPGA implementation. They accomplish 560 clock cycles/MB at a clock frequency of about 140 MHz to process 1080p in real time. The implementation uses a quite old technology (130 nm). In [10] a codec architecture is given, which performs about the same, but in contrast to the previous mentioned implementations besides the ASIC design an FPGA implementation is described. Authors in [15, 19, 34] introduce very similar architectures, all having in common that they are based on a MB processing pipeline. There are

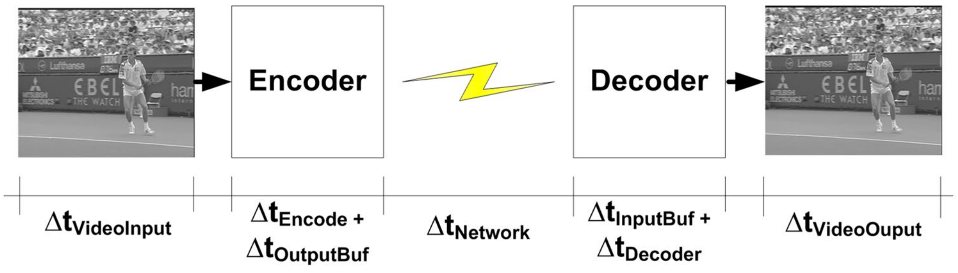

Figure 3 Causes of video coding latency.

two main advantages of this approach. First, this processing scheme keeps the amount of buffer memory needed small, which is very important for hardware implementations in general. Second, MB-based processing guarantees deterministic throughput numbers, something we strive for, especially in terms of low-latency processing. The problem with these approaches is, that the obtained data rates are quite high, which prevents us from using bandwidth limited wireless communications channels. To attain a better efficiency it is strongly desired to use all other available coding tools provided by the H.264 video coding standard. In particular, the use of inter predictive coding modes raises the coding efficiency significantly, but as described in [38] it considerably increases the complexity. In [8, 9, 16, 22–25, 29, 39] a couple of dedicated hardware implementations for H.264 codecs supporting at least P- and in most cases also B-predictive coding modes are described. Another common feature of all these implementations is that they are not optimized towards low-latency processing, since the rate control algorithms in the codecs rely on huge buffer sizes in order to relax mode decisions. Common for all of the refereed implementations is the fact, that they have been implemented as dedicated hardware blocks for a given semiconductor technology. Fabricating a dedicated chip for the described application field was not possible due to limited amount of funding and project duration. We have designed our codec for an FPGA target platform instead. This solution can hardly be compared to dedicated chip design. Various FPGA related building blocks for H.264 video encoders can be found in the literature. In [27, 28, 31] the design space for H.264 quantizers/dequantizers for FPGAs are discussed, which is also suitable for H.264 video decoders. An architecture for intra predictions is discussed in [41]. Other building blocks such as transform, entropy encoders or decoders and motion estimation can be found in [17, 18, 26, 33]. In [4] a codec architecture based on processor cores implemented on an FPGA is given. It results in similar performance points as processor based software implementations. In [29] results for a complete H.264 encoder is given, which has been developed as a dedicated hardware IP, but mapped to an FPGA for hardware verification purposes.

To the best of our knowledge, the presented codec architecture (encoder and decoder) is the only complete non-commercial H.264 implementation that provides all features for technical encoding applications with intra and inter encoding, very low latency even in inter mode, guaranteed rate control and ROIs.

3 Low Latency Video Codec

3.1 Encoder

The H.264/AVC baseline profile compliant video encoder [14] is designed around a 5-stage MB pipeline as shown in Fig. 4 to encode incoming video data with a near constant bit rate in real-time for formats up to level 4.1 (e.g. 1920x1080p30). The encoder is entirely implemented in hardware with vendor-independent VHDL (Very High Speed Integrated Circuit Hardware Description Language).

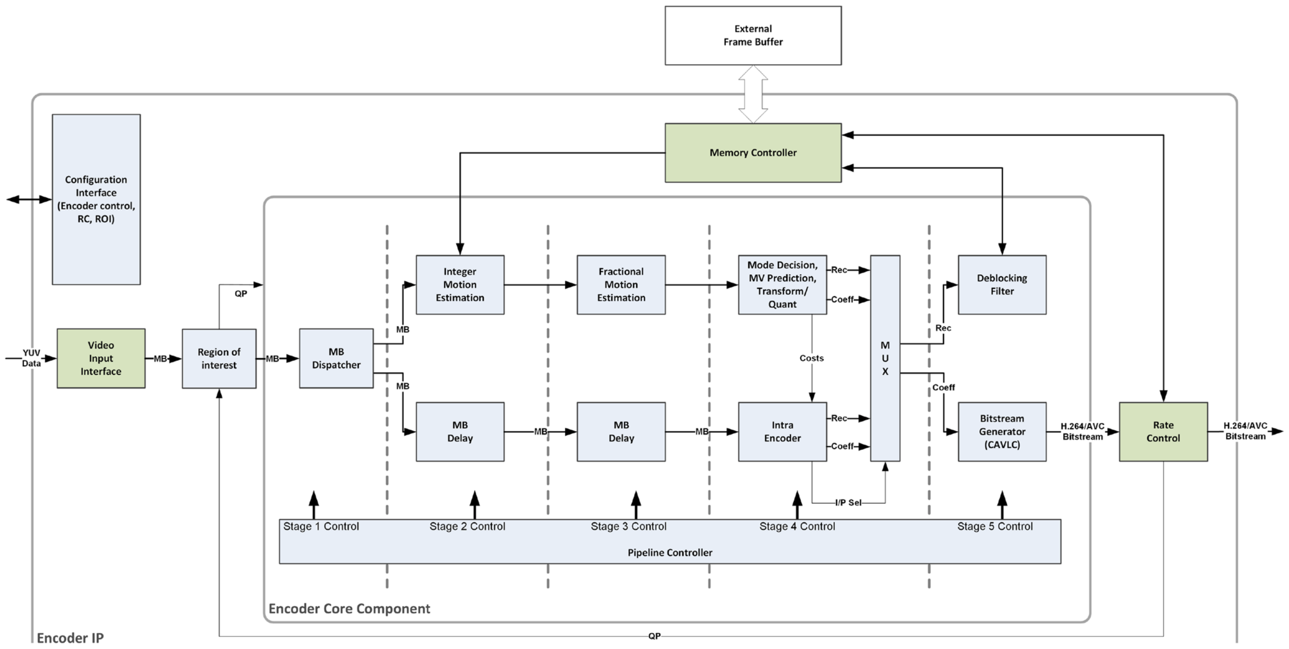

Figure 4 Encoder 5 stage pipeline block diagram.

No processor is used. Each MB is processed within a stage with a constant delay of 613 clock cycles, which is a direct result of the maximum number of cycles specified by hardware description to calculate one MB. Therefore the minimum needed clock frequency for real-time encoding is calculated as follows:

$$\mathit{System\ clock}_{\mathit{Encoder}} = \mathit{Width\ in\ MBs} \cdot \mathit{Height\ in\ MBs} \cdot 613 \cdot \mathit{framerate} \quad (2)$$

If the system clock frequency is reached, then real-time processing is ensured and the requirement R1 is fulfilled. An HD video can thus be processed in real-time at a system clock of 150.1 MHz or higher.

In stage 1 the Video Input Interface converts the incoming 4:2:0 YCbCr 8 bit video stream into a MB format. This way, a sub-marcoblock (SMB) can always be processed in only one external memory burst and the bandwidth can be used efficiently. After image data and quantization parameter (QP) have been adjusted according to the set ROIs (details in Sect. 3.1.2), the MBs are distributed to both the intra- and the inter-pipeline. In the inter-pipeline, motion estimation is first performed with pixel accuracy at stage 2 (Integer Motion Estimation, IME), then with sub-pixel accuracy at stage 3 (Fractional Motion Estimation, FME) with a search window of $\pm$ 13 pixels. Within the search window the motion vectors are unrestricted. Finally, in stage 4 the Chroma Motion Estimation together with the Mode Decision and the Motion Vector Prediction is performed. In contrast, in the parallel intra-pipeline the stages 2 and 3 are only used for delay. The Intra Encoder in stage 4 accomplishes the intra prediction and transformation of the video MB by MB. It supports all 4x4 intra modes as well as 16x16 vertical, horizontal and DC modes. Plane Prediction is not supported, as it provides only 1% coding efficiency while almost doubling the resources required for intra-coding. On MB basis, either the residual data of the intra (I) or inter (P) pipeline are taken as basis for generating a baseline compliant bitstream using a Context Adaptive Variable Length Coding (CAVLC) entropy encoder in stage 5. At this point the bit rate is very variable. Also in stage 5, the reconstructed MBs are written to external memory as reference frame. Before that, the MBs are filtered by an in-loop deblocking filter (DBF) to reduce blocking artifacts. After stage 5, the rate control block ensures a constant target bit rate. The encoder does not support the encoding of B-frames despite possible higher coding efficiency, since this would increase the latency by using succeeding images for prediction. This contradicts requirement R4. Furthermore, it would raise the complexity of the design which contradicts requirement R5. The behavior of the MB pipeline across the stages is shown in the Fig. 5.

3.1.1 Rate Control

In order not to exceed a constant target bit rate under all circumstances and thus meet the requirement R2, we combine two different approaches. An encoder without rate control generates a variable bit rate with a fixed QP ranging from 0 to 51, where 51 results in very low image quality but a low bit rate. To generate a bitstream with constant data rate the fixed QP has to be adjusted on MB-basis. Rate control should optimize the QP so that the produced bit rate fits the target bit rate as well as possible and on the other hand achieves the best picture quality at that bit rate. Since the bit rate depends on the image content and can therefore vary considerably, a large number of QP changes would be inevitable even within a frame. The overall subjective quality would suffer from these many QP changes and distortions would occur even at higher target bit rates. Hence the strategy for rate control should be:

- Obtain the target bit rate, as precise as possible.
- Choose a QP as low as possible to achieve best quality.
- Try to minimize variations of QP within one frame and from frame to frame whenever possible.

To achieve a constant bit rate with this strategy a bitstream buffer is required, which is written variably and read out constantly with the target bit rate according to the FIFO principle to compensate bit rate variations. This buffer

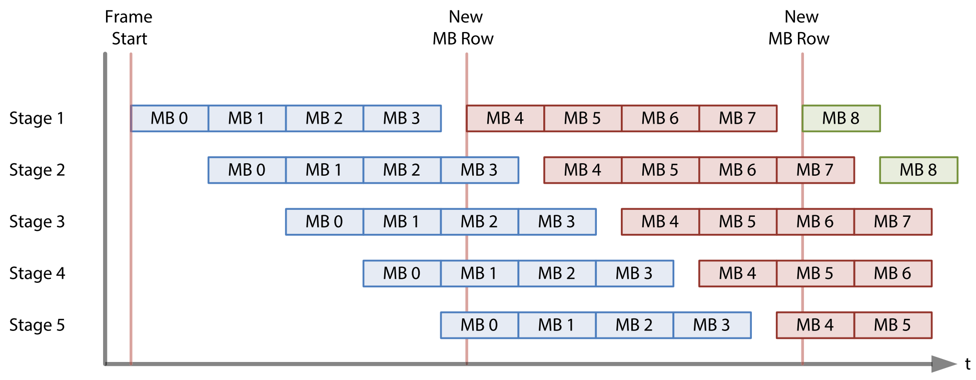

Figure 5 Encoder MB pipeline behavior.

is depicted in Fig. 6. Depending on the desired size, the buffer can be implemented in the internal block RAM or in an external memory. If the size of the output buffer is rather large, more variations can be equalized within this buffer than, when using a small buffer, so rate control can allow more variations of the bit rate. The size of this output buffer defines the overall end-to-end latency from encoder to decoder so that for low-latency applications the buffer size is limited. Since the rate control is not in all cases able to hit the target bit rate, variations are allowed up to a certain overall limit. This means, that we have combined CBR and VBR towards a capped VBR rate control strategy.

Algorithm 1 Calculate QP.

```
Require: $N, M, fps, tb, cb, bl, QP_{l1}, QP_{l2}, fs_{l1}, fs_{l2}$
  $QP \leftarrow f(N, M, fps, b)$
  while new frame do
      if $bl > 80\ \%$ then
          $QP \leftarrow QP + 2$
      else if $bl > 60\ \%$ then
          $QP \leftarrow QP + 1$
      else if $bl < 12\ \%$ then
          $QP \leftarrow QP - 2$
      else if $bl < 25\ \%$ then
          $QP \leftarrow QP - 1$
      else
          $QP \leftarrow QP$
      end if
      for all macroblocks do
          if $bl > 87.5\ \%$ then
              $QP \leftarrow QP + 1$
          else
              $QP \leftarrow QP$
          end if
      end for
      $QP_{change} \leftarrow f(cb, tb, QP_{l1}, QP_{l2}, fs_{l1}, fs_{l2})$
      if $cb > tb$ then
          State $QP \leftarrow QP + QP_{change}$
      else
          State $QP \leftarrow QP - QP_{change}$
      end if
  end while
```

Primary task of the rate control is to regulate the QP based on the current value of the output buffer fullness so that no buffer over- or underflow occurs by always following the guidelines below (see Algorithm 1):

- At the start of a new sequence, the initial *QP* is calculated based on the image resolution *N* and *M*, the frame rate *fps* and the target bit rate *tb*, since no information about the video footage is available. Because the first frame is always intra encoded, the *QP* must be set high enough (e.g. 45 for HD@12MBit) so that the bitstream buffer does not overflow. When the target bit rate has a reasonable value, the start *QP* decrements very fast and so the image quality increases within some few frames.
- At the start of each frame, the calculated *QP* is modified according the determined buffer-level *bl* to ensure a proper buffer fullness around 50%. When this level reaches a threshold value of 60% or 80%, the *QP* is incremented by one or two respectively to prevent a buffer overflow. If the buffer has a low filling level below 12% or 25%, the *QP* is decremented by two or one respectively. The calculated *QP* is used for encoding the next frame.
- After coding a MB, the buffer level *bl* is checked. If a critical value of 87.5% is reached, the *QP* used for the next MB is increased by one. If the selected buffer size is quite small due to a hard requirement for low latency, more regulation is needed to ensure an adequate buffer level.
- After each frame, the frame bit size is checked and adapted. According to the used QP, the size of the coded frame *cb* will differ from the target frame *tb* size more or less. If the difference is within ± 10% the QP is not changed, to minimize distortion based on QP changes from frame to frame. If the difference is larger, QP for the next frame will be adapted based on the difference to the target size and also on basis of the QPs of the two previous frames QP_{l1}, QP_{l2} and their frame sizes fs_{l1}, fs_{l2}.

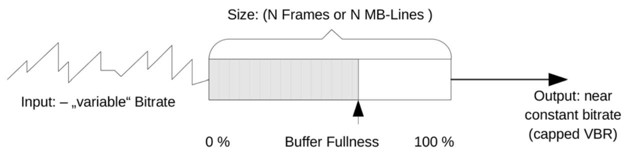
Size: (N Frames or N MB-Lines)
Input: – „variable" Bitrate
0 %
Buffer Fullness
100 %
Output: near constant bitrate (capped VBR)

Figure 6 Bitstream output buffer.

Normally, an encoded video consists of a sequence of I- and P-frames, where I-frame and especially IDR-frames serve as synchronization point for the decoder. Intra frames require a higher bit rate than P-frames because no motion estimation is used. This leads either to peaks in the required bandwidth or to quality degradation for each I-frame. Both effects are undesirable. Therefore, apart from the very first frame of a stream, we only use P-frames with embedded stripes of I-MBs as shown in Fig. 7. The complete intra update of the MBs is now done by several successive P-frames, in the shown example 6 frames are needed. The intra refresh mode (IRM) avoids the problem of large intra encoded frames, because the high bit rate required for an I-frame is divided among the P-frames. To allow a decoder to still detect a synchronization point in the bitstream, we use Supplemental Enhancement Information (SEI) message, which provides the information how long a refresh cycles takes and when to start displaying picture content. Receiving a recovery point SEI message automatically prevents the decoder from waiting for an I/IDR-frame and prompts it to immediately start decoding the received P-frames. As a

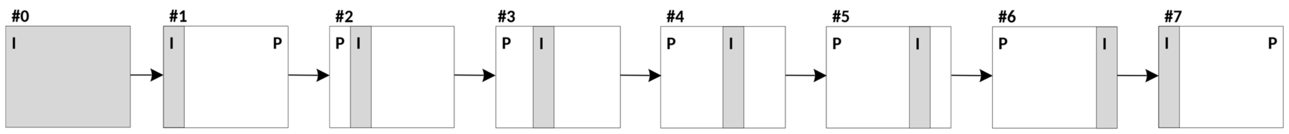

Figure 7 Propagation of the Intra refresh column throughout a video sequence.

result of the more balanced bit rate in IRM the rate control buffer can be smaller and generates less latency, thus fulfilling the requirement R4.

3.1.2 Region of Interest

To ensure that the image quality is sufficiently good for shipwreck detection even after coding with limited bandwidth, the encoder uses a number of ROI to improve the image quality of interesting image areas. Within the ROI the MB are encoded with a lower QP. Therefore the image quality is higher there. Each ROI is defined as a rectangle over a start position in X and Y direction plus the ROI width and height. The coordinates must be set from outside the encoder. The positioning of the ROI can either be done manually or via a simple DCNN for object detection like YOLO [30].

When using ROI, a constant bit rate should still be guaranteed (see Sect. 3.1.1). This is necessary for communication channels of limited bandwidth. To achieve this, we define a QP difference, by which the QP outside the ROIs is increased, meaning a poorer image quality at a lower total bit rate. Due to the lower bit rate the rate control notices that the target bit rate is no longer reached. The QP is lowered and thus the bit rate in the ROIs is increased. After a few frames, the control loop reaches the target bit rate again despite the increased QP outside the ROIs. Thus the image quality within the ROIs is increased and requirement R3 is met. Since not only the ROIs are transferred, the complete image is still available in lower quality for manual analysis as required in R6. To further reduce the required bit rate, the region outside the ROIs can be encoded without the chroma channels. The feedback loop then further increases the quality within the ROIs.

3.1.3 External Memory Requirements

The temporal dependencies of P-frames makes it necessary to store whole uncompressed frames in memory so that they can be accessed later (i.e. reference frames). Since the size of the internal FPGA BRAM (Block RAM) is not sufficient for this task, external memory is utilized. In order to reduce the number of accesses needed the reference frame is stored as SMB with 4x4 pixels in the memory. Figure 4 shows that the encoder requires three memory operations:

- read the luma search window (SW) data and read the chroma SW data for Motion Compensation (both through the IME block)
- read a part (lower SMB) of the adjacent upper MB that could not be kept locally for the DBF and write back the reconstructed MB.
- write the bitstream to the Rate Control buffer and read from it.

These operations require a continuous memory of at least 128 MB. The memory has to fulfill specific throughput requirements in order to successfully store the reconstructed MB and read the necessary MBs of the active reference frame for inter prediction. This requirement results from the following formula:

$$TP_{EncTotal} = TP_{IME} + TP_{MotionCompensation} + TP_{WriteDBF} + TP_{ReadDBF} + TP_{RateControl} \tag{3}$$

$$TP_{IME} = 256Byte/MB \cdot MBs/SWmax \cdot MBPF \cdot FPS \tag{4}$$

$$TP_{MotionCompensation} = 256Byte/MB \cdot Chr_{factor} \cdot MBs/SWmax \cdot MBPF \cdot FPS \tag{5}$$

$$TP_{WriteDBF} = (64Byte/TopMB + 256Byte/LeftMB) \cdot (1 + Chr_{factor}) \cdot MBPF \cdot FPS \tag{6}$$

$$TP_{ReadDBF} = 64Byte/MB \cdot (1 + Chr_{factor}) \cdot MBPF \cdot FPS \tag{7}$$

$$TP_{RateControl} = 2 \cdot bitratepersecond \tag{8}$$

where:

- MBs/SW max is a worst case number of MBs that needs to be read to refresh the SW, e.g. 6 in the case of $\pm$ 13 pixels search range.
- MBPF is the number of MBs per frame.
- FPS is the frame rate of the processed video.
- $Chr_{factor} = 2$ for the chroma channels because the internal format is 4:4:4 YCbCr.

For an HD image (with 8160 MBs) at 25 fps, the formulas result in a required rounded up throughput of:

$$TP_{EncTotal} = 314MBps + 627MBps + 196MBps + 40MBps + 20MBps = 1197MBps \tag{9}$$

3.2 Decoder

The fully baseline profile compliant H.264/AVC decoder includes three MB-based pipeline stages for entropy decoding, coefficient reconstruction and in-loop filtering. The decoder is implemented as vendor-independent VHDL completely in hardware and without using a processor. Each stage is decoupled by a MB double buffer as shown in Fig. 8. Each MB is processed in a maximum of 640 clock cycles, which again is directly derived from the hardware description of the decoder. To ensure real-time processing, the minimum required system clock is calculated as follows:

$$System\ clock_{Decoder} = Width\ in\ MBs \cdot Height\ in\ MBs \cdot 640 \cdot framerate \tag{10}$$

For the HD format with 30 fps this results in a minimum clock frequency of 156.7 MHz. Above this frequency requirement R1 for the decoder is satisfied.

The bitstream interface is connected to the Variable Length Decoder (VLD) component in stage 1, which is used for CAVLC entropy decoding. All necessary syntax elements for intra and inter reconstruction, e.g. intra prediction modes and inter motion vectors, are stored in a MB double buffer, which can be accessed by the subsequent pipeline stage. The second pipeline stage (stage 2) contains the Intra Prediction (IPR), Inter Motion Compensation (MC) and Residual Generation (RES), which form the intra and inter reconstruction part of the decoder. Output of this stage is a completely reconstructed MB with unfiltered coefficients. The third pipeline stage (stage 3) contains an in-loop DBF, which is used to improve the visual quality. Picture samples are filtered with respect to current and adjacent MB parameters and already filtered neighbor samples. The decoded video is output as 8 bit 4:2:0 YCbCr. The AXI master component (AXI bus protocol [1]) processes all write and read requests of the connected components and drives the AXI master interface.

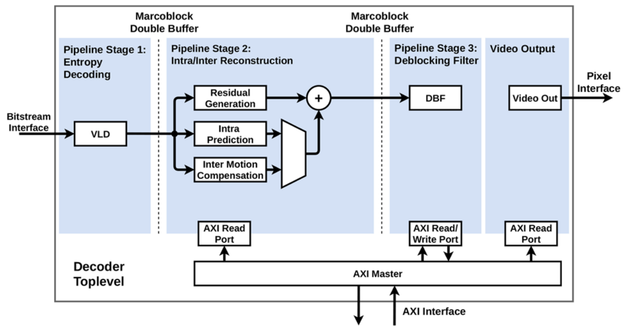

Fig. 8 Decoder 3 stage pipeline block diagram

The decoder needs 107 MB to store a maximum of 17 reference frames. To reduce the number of memory accesses required, the reference data is stored in SMBs in the external memory, in the same way as in the encoder. Three memory operations take advantage of these reference frames:

– read Motion Compensation data. For each vector processed it is necessary to read a 9x9 pixel block containing all pixel needed for interpolation from the memory. Therefore 12x12 pixels (3x3 SMBs) will be read for each SMB processed.
– read the lower SMB of the adjacent upper MB and write back the reconstructed MB (like in the encoder).
– read video output data.

$$TP_{DecTotal} = TP_{MotionCompensation} + TP_{WriteDBF} + TP_{ReadDBF} + TP_{VideoOutput} \tag{11}$$

$$TP_{MotionCompensation} = 16Byte/SMB \cdot (1 + Chr_{factor}) \cdot 9 \cdot MBPF \cdot 16 \cdot FPS \tag{12}$$

$$TP_{WriteDBF} = (64Byte/TopMB + 256Byte/LeftMB) \cdot (1 + Chr_{factor}) \cdot MBPF \cdot FPS \tag{13}$$

$$TP_{ReadDBF} = 64Byte/MB \cdot (1 + Chr_{factor}) \cdot MBPF \cdot FPS \tag{14}$$

$$TP_{ReadVideoOutput} = 256Byte/MB \cdot (1 + 0.5) \cdot MBPF \cdot FPS \tag{15}$$

where:

– MBPF is the number of MBs per frame.
– FPS is the frame rate of the processed video.
– $Chr_{factor} = 0.5$ for the chroma channels.

For an HD image (with 8160 MBs) at 25 fps, the formulas result in the rounded up throughput of:

$$TP_{DecTotal} = 705.1MBps + 98MBps + 19.6MBps + 78.4MBps = 901.1MBps \tag{16}$$

3.3 Codec Latency

For the calculation of the overall codec latency Δt_{all} according to formula 1 for a 1920x1080p25 sequence (with 120 MB per line) we have to note that $\Delta t_{VideoInput}$ and $\Delta t_{VideoOutput}$ depend on the camera respectively monitor used and their connection. The delay of the transmission channel $\Delta t_{Network}$ is also unknown and typically varies significantly. Therefore

Table 2 A10 device family encoder resource utilization with 160 MHz clock constraint (rounding errors possible)

	ALM	%	Memory bits	%	M20K BRAM	%	DSP	%
Video Input	1109	1.8	802816	54.1	58	12.9	0	0
ROI	289	0.5	0	0	0	0	0	0
Intra pipeline	7553	12.4	223104	15.0	119	26.5	33	50.8
Inter pipeline	38180	62.5	310748	21.0	188	41.9	32	49.2
DBF	6022	9.9	51876	3.5	45	10.0	0	0
CAVLC	4687	7.7	13904	0.9	16	3.6	0	0
Rate Control	2048	3.4	13056	0.9	9	2.0	0	0
Control	648	1.1	0	0	0	0	0	0
Memory IF	567	0.9	67312	4.5	14	3.1	0	0
Overall	61104	14.3	1482816	2.7	449	16.5	65	4.3

we do not specify these delays. The remaining latency is fixed due to the MB pipeline and does not vary. Using larger buffers results in higher latency. The remaining delays can be determined as a minimum latency example at a clock rate of 160 MHz for both the encoder and the decoder as follows:

- The double buffer at the input stores one MB line and therefore adds 120 MB latency. Each of the 5 encoder stages of the MB pipeline is adding another MB latency. As a result the minimum possible encoder delay is

$$\Delta t_{Encode} = \frac{(120 + 5MB) \times 613 cycles/MB}{160MHz} = 0.47ms \tag{17}$$

- The minimal output buffer size is one MB line. For 25 fps, this results in a latency of

$$\Delta t_{OutputBuf} = \frac{40ms}{8160MB} \times 120MB = 0.59ms \tag{18}$$

- $\Delta t_{InputBuf}$ should be as large as $\Delta t_{OutputBuf}$ so the latency is the same.
- The 3 decoder stages means 3 MB delay, so that

$$\Delta t_{Decode} = \frac{(3MB) \times 640 cycles/MB}{160MHz} = 0.012ms \tag{19}$$

The total minimum delay is therefore:

$$\begin{aligned}\Delta t_{all} = \Delta t_{VideoInput} + 0.47ms + 0.59ms \\ + \Delta t_{Network} + 0.59ms + 0.012ms + \Delta t_{VideoOutput} \\ = 1.662ms + \Delta t_{VideoInput} + \Delta t_{Network} \\ + t_{VideoOutput}\end{aligned} \tag{20}$$

4 Results

Tables 2 and 3 show the resource utilization of the encoder on two different Intel FPGAs, the low-end FPGA 10CX220YF780E5G (C10) and the mid-range FPGA 10AX115N3F40E2SG (A10). Tables 4 and 5 present the resources of the decoder for the same FPGAs. We have tested the encoders and decoders in hardware on the A10 FPGA, the numbers for C10 represent pure fitting results. We performed the synthesis and place and route process using Quartus 19.4. For the encoder we used the balanced setting, for the decoder the high performance effort setting to achieve better timing. With [20] we have found only one comparable AVC encoder with inter prediction implementation for FPGAs. It targets an older version of our Arria 10

Table 3 C10 device family encoder resource utilization with 160 MHz clock constraint (rounding errors possible)

	ALM	%	Memory bits	%	M20K BRAM	%	DSP	%
Video Input	1118	1.9	802816	53.8	58	12.2	0	0
ROI	257	0.4	0	0	0	0	0	0
Intra pipeline	7437	12.4	225920	15.1	125	26.2	33	50.8
Inter pipeline	37648	62.5	316388	21.2	207	43.4	32	49.2
DBF	5877	9.8	52260	3.5	48	10.1	0	0
CAVLC	4671	7.8	13904	0.9	16	3.4	0	0
Rate Control	2021	3.4	13056	0.9	9	1.9	0	0
Control	606	1.0	0	0	0	0	0	0
Memory IF	562	0.9	67312	4.5	14	2.9	0	0
Overall	60198	74.9	1491656	12.5	477	81.3	65	33.9

Table 4 A10 device family decoder resource utilization with 160 MHz clock constraint (rounding errors possible)

	ALM	%	Memory bits	%	M20K BRAM	%	DSP	%
VLD	12568	38.3	318752	63.0	43	32.8	2	10.5
Video Out	908	2.8	8192	1.6	8	6.1	0	0
IPR	3068	9.3	57344	11.3	11	8.4	1	5.3
RES	2044	6.2	5760	1.1	6	4.6	16	84.2
MES	7416	22.6	29040	5.7	10	7.6	0	0
DBF	3338	10.2	52664	10.4	33	25.2	0	0
Control	2730	8.3	0	0	0	0	0	0
AXI IF	757	2.3	34404	6.8	20	15.3	0	0
Overall	32831	7.7	506156	0.9	131	4.8	19	1.3

family (20 nm process) by using the Arria II family (40 nm) and achieves a base clock of 100 MHz with 200 MHz in some parts. The design requires on average 652 clock cycles per MB in the worst case (Inter MB with intra 4x4 prediction), which is slightly more than 613 cycles for our encoder, which moreover never exceeds this limit. According to the author, a throughput of 1080p30 is achieved. Following formula 2, this requires a minimum clock of 159.6 MHz for the MB pipeline, so the pipeline seems to work with 200 MHz. With 77k ALUTs (Adaptive Look-Up Tables) this design requires less ALUTs than our design with 132k ALUTs. The amount BRAM used is with 148.6 kB slightly smaller than our implementation for the A10 with 185.4 kB. With 66 DSPs [20] uses one DSP more than we do. The specified power consumption refers to the ASIC (application-specific integrated circuit) implementation and is therefore not comparable. However, technical parameters such as very low latency and a data rate guaranteed under all circumstances as well as the possibility of defining ROIs, which the encoder in [20] does not offer, are particularly relevant for the presented application.

The percentage in the row Overall refers to the total resources of the FPGA, the other percentages to the share in the IP core. In the Tables 2 and 3 it can be seen that a large part of the ALMs (Adaptive Logic Module) and the memory bits are used by the inter pipeline. However, the coding gain still justifies the use of P-frames. In contrast, capabilities such as Rate Control and especially ROIs require few resources. In addition, it is evident that more internal RAM blocks (M20K) are used in percentage terms than memory bits. Obviously, the memory sizes used do not map well to the 512 x 40 bit block RAM structures of the used FPGAs. Since this is the bottleneck of the encoder implementation, an optimization of this mapping seems reasonable. In contrast, as depicted in the Tables 4 and 5, the ALMs are the bottleneck in the decoder. However, the decoder already requires significantly fewer resources.

As shown in Table 6, even on a low-end FPGA like the C10 it is possible to run either the encoder or the decoder with a sufficient clock frequency for processing HD images at 30 fps, proving the low complexity (requirement R5). Since the resource utilization by other parts like video IO or external memory depends on the board, we have not specified them in the resource utilization tables. But due to the increased resource usage it is possible that a Cyclone FPGA is not sufficient anymore. The mid-class FPGAs of the Arria family such as 10AX115N3F40E2SG offer enough space for the peripherals. With an assumed maximum utilization of 70%, it is even possible to implement 4 encoders or 8 decoders together as the numbers in the Tables 2 and 4 prove. In Table 6, we also listed the power requirements of the IP cores at 160 MHz target clock. The calculation was performed with an assumed toggle rate of 12.5 %. The higher power dissipation of the A10 results solely from the higher static power dissipation due to the larger FPGA.

Table 5 C10 device family decoder resource utilization with 160 MHz clock constraint (rounding errors possible)

	ALM	%	Memory bits	%	M20K BRAM	%	DSP	%
VLD	12402	37.9	318752	63.0	43	32.8	2	10.5
Video Out	905	2.8	8192	1.6	8	6.1	0	0
IPR	3073	9.4	57344	11.3	11	8.4	1	5.3
RES	2048	6.3	5760	1.1	6	4.6	16	84.2
MES	7488	22.9	29040	5.7	10	7.6	0	0
DBF	3336	10.2	52664	10.4	33	3.4	0	0
Control	2718	8.3	0	0	0	0	0	0
AXI IF	780	2.4	34404	6.8	20	15.3	0	0
Overall	32749	40.8	506156	4.2	131	22.3 %	19	9.9

Table 6 Codec maximum clock and power dissipation (at 160 MHz clock)

	Device	Clock	Power
Encoder	10CX220YF780E5G	164.8 MHz	2647.9 mW
	10AX115N3F40E2SG	204.4 MHz	3860.3 mW
Decoder	10CX220YF780E5G	180.7 MHz	1415.6 mW
	10AX115N3F40E2SG	192.5 MHz	2483.9 mW

In Fig. 9 is shown how the Y PSNR value for different videos sequences (1920x1080@25fps) changes with the achieved bit rate for the latency of one frame, which is slightly below the constant target bit rate. As the bit rate increases, the quality of the encoded video also increases, which is expressed by a higher PSNR. With the three different sequences, the different PSNR show the influence of the video content on the quality at the same target bit rate. A sequence with high motion like Crowdrun achieves a significantly lower PSNR at a similar bit rate compared to a sequence with little motion like Sunflower. In Fig. 10 we look at the influence of ROIs on the image quality of the sequence with the lowest quality. The PSNR curve in the middle (+) shows a coding without ROIs as reference. The influence of four ROIs with 224x224 pixels each (approx. 10% of the overall HD image region) is shown by the other two curves. The upper curve (∗) shows the PSNR only for the ROIs and the lower curve (×) for the complete image including ROIs. Our mechanism for treating interesting areas allows us to achieve a significantly higher quality (around 9 dB better) within these areas compared to a sequence without ROI. But since the bit rate is constant, this is achieved at the expense of the quality of the rest of the image. There the PSNR is about 3 dB lower than without ROIs.

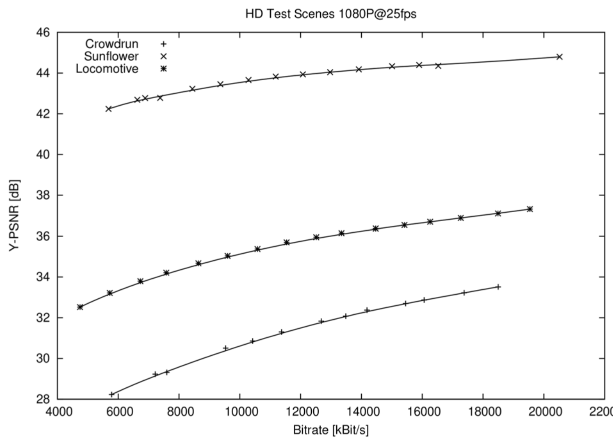

Figure 9 Y PSNR for different HD@25fps scenes with IRM.

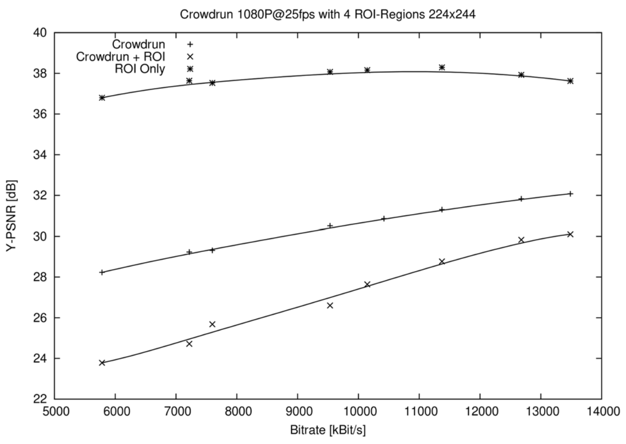

Figure 10 Y PSNR for Crowdrun (HD@25fps) with IRM and 4 ROIs.

5 System Integration

In the hardware design, the communication interfaces, the internal bus system and the interfaces to the off-chip memory must be implemented again for each board. To simplify this process and to use reliable components, hardware frameworks are used. We use the framework proposed by the authors in [37] to make our H.264/AVC video codec easily available over a standardized network interface as Network-attached Accelerator (NAA). We configured the framework as shown in Fig. 11 with two sockets allowing two independent accelerators, e.g. an encoder and a decoder to be operated in parallel. The video data to be encoded from the camera (1 in Fig. 1) is streamed over a 40 Gbps network interface to the encoder. There, the stream is encoded, packed into a transport stream (TS) [13] and sent to the transmitting unit of the aerial link (3 in Fig. 1). On the decoder side, the TS is streamed into the accelerator via the network interface, unpacked from the TS and decoded. The decoded video or the ROIs only are sent via the network interface to a ML component (6 in Fig. 1). This pipeline is well suited for embedded integration due to its

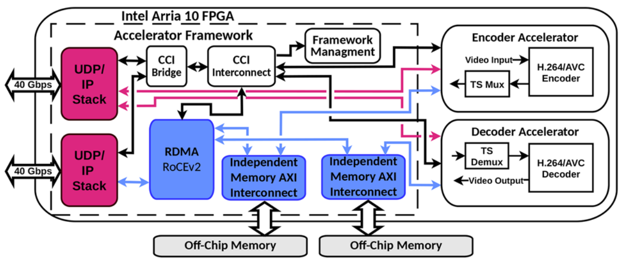

Figure 11 Network-attached accelerator with two sockets.

Table 7 Network-attached Accelerator resource utilization (rounding errors possible). Percentages refer to the available resource amount

	ALM	%	Memory bits	%	M20K BRAM	%	DSP	%
UDP Stacks	10270	2.4	240352	0.4	64	2.4	0	0
RoCEv2 Stack	13534	3.2	1192	0.2	58	2.1	0	0
Memory Interconnect	3868	0.9	575280	1.0	60	2.2	0	0
CCI Interconnect	923	0.2	2528	0	6	0.2	0	0
Framework Management	601	0.1	432	0	4	0.1	0	0
Framework	**29196**	**6.8**	**937872**	**1.7**	**192**	**7.1**	**0**	**0**
DDR Controller (ECC)	8325	1.9	217472	0.4	42	1.5	0	0
MAC	19044	4.5	1093888	2.0	80	2.9	0	0
Framework+Controller	**56565**	**13.2**	**2249232**	**4.0**	**314**	**11.6**	**0**	**0**
Encoder+Wrapper	68792	16.1	1801056	3.2	487	18.0	66	4.3
TS Mux	1003	0.2	434176	0.8	29	1.1	0	0
Decoder+Wrapper	40926	9.6	648176	1.2	151	5.6	20	1.3
TS Demux	206	0	256	0	2	0.1	0	0
Overall	**167492**	**39.2**	**5132896**	**9.2**	**983**	**36.2**	**86**	**5.7**

Bold emphasis represents the sum of the previous lines up to and including the prior bold emphasis

simplicity and clear separation. In addition, the hardware framework and the associated software framework can be used in the data center environment. The data is exchanged via the off-chip memory using Remote DMA via RoCEv2 (RDMA over Converged Ethernet). This setup serves us as a test application.

The resource usage of the framework components is shown in Table 7. Compared to the resource consumption of the stand-alone encoder/decoder in the previous section, the encapsulated components require more resources, mainly due to clock domain crossing (CDC) to the interfaces of the framework. As illustrated, the framework leaves the majority of resources free for the accelerators as it uses only 13.2 % of the ALMs (Adaptive Logic Module), 11.6 % of the M20K Block RAMs and 0 DSPs. By omitting features such as error-correcting code (ECC) and RoCEv2 or by using only one network interface, further resources can be saved if necessary.

6 Conclusion and Future Work

In this paper we have described the architecture of a low complexity H.264/AVC video codec, which has been optimized regarding its usage in technical image processing applications. We have focused on two major issues arising from using video compression in these applications, which are mainly providing low latency and avoiding compression artifacts. By offering the possibility to preserve particular image regions to be encoded with higher quality demands compared to the remaining image, we have found an interesting way to combine video coding efficiency with the requirements of sophisticated image processing algorithms such as DCNNs. Besides being a low-complexity implementation paving the way for using mid-range or even low-end FPGAs, we were able to fulfill the demanding low latency requirements while still achieving moderate coding efficiency and constant bit rate for using rate constrained wireless links. Combining the IRM with an optimization of the needed input/output buffers and our rate control leads to sub frame latencies. Both parts of the video codec, encoder and decoder, have been verified against the official ITU [2] reference models and successfully implemented on FPGAs. They will be used in the described applications scenario given in Sect. 1. By integrating the video codec into a hardware framework and its usage as NAA, we have demonstrated a straightforward way to deploy it in both an embedded and data center environment.

Our future work will be focused on two main areas. At first a comprehensive testing of the overall system needs to be conducted, especially incorporating our DCNNs, which will then have been trained for the recognition of shipwrecked persons. This has not been done yet, since labeling of the recorded video data is still taking place. Another broad research topic is the enhancement of our low-complexity rate control algorithm to enhance the achieved coding efficiency of the codec. Since we have developed an H.265/HEVC decoder, which has been described in [35], we will work on the integration of our concepts into a HEVC low latency video codec architecture for high coding efficiency as well.

Acknowledgements We thank Prof. Dr. Andreas Zell from the Chair of Cognitive Systems at the University of Tübingen for providing the background water image in Fig. 1. This work has been partly developed in the AVALON project which is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) with the reference number 03SX481E.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. AMBA AXI and ACE protocol specification (2014). [Accessed October 27th, 2020] http://infocenter.arm.com/help/topic/com.arm.doc.ihi0022d/index.html
2. H.264.2: Reference software for ITU-T H.264 advanced video coding (2017). [Accessed October 27th, 2020] https://www.itu.int/rec/T-REC-H.264.2-201602-I/en
3. Avgousti, S., Panayides, A. S., Jossif, A. P., Christoforou, E. G., Vieyres, P., Novales, C., Voskarides, S., & Pattichis, C. S. (2016). Cardiac ultrasonography over 4g wireless networks using a tele-operated robot. *Healthcare Technology Letters, 3*(3), 212–217. https://doi.org/10.1049/htl.2016.0043
4. Belhadj, N., Turki, M., Marrakchi, Z., Ben Ayed, M. A., Masmoudi, N., & Mehrez, H. (2013). MPSOC architecture for h.264/AVC intra prediction chain on SOCLIB platform and FPGA technology. In: *14th International Conference on Sciences and Techniques of Automatic Control Computer Engineering - STA'2013* (pp. 216–219). https://doi.org/10.1109/STA.2013.6783133
5. Castro, C. A., Alqassis, A., Smith, S., Ketterl, T., Sun, Y., Ross, S., Rosemurgy, A., Savage, P. P., & Gitlin, R. D. (2013). A wireless robot for networked laparoscopy. *IEEE Transactions on Biomedical Engineering, 60*(4), 930–936. https://doi.org/10.1109/TBME.2012.2232926
6. Chen, J., Chang, H., Wang, J., & Guo, J. (2011). A dynamic quality-adjustable h.264 intra coder. *IEEE Transactions on Consumer Electronics, 57*(3), 1203–1211. https://doi.org/10.1109/TCE.2011.6018875
7. Chen, X., Hwang, J., Lee, K., & de Queiroz, R. L. (2015). Quality-of-content (GOC)-driven rate allocation for video analysis in mobile surveillance networks. In: *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)* (pp. 1–6).
8. Chen, Y., Chen, T., Tsai, C., Tsai, S., & Chen, L. (2009). Algorithm and architecture design of power-oriented h.264/avc baseline profile encoder for portable devices. *IEEE Transactions on Circuits and Systems for Video Technology, 19*(8), 1118–1128. https://doi.org/10.1109/TCSVT.2009.2020323
9. Ding, L., Chen, W., Tsung, P., Chuang, T., Hsiao, P., Chen, Y., Chiu, H., Chien, S., & Chen, L. (2010). A 212 mpixels/s 4096 x 2160p multiview video encoder chip for 3d/quad full HDTV applications. *IEEE Journal of Solid-State Circuits, 45*(1), 46–58. https://doi.org/10.1109/JSSC.2009.2031787
10. Diniz, C., Zatt, B., Thiele, C., Susin, A., Bampi, S., Sampaio, F., Palomino, D., & Agostini, L. (2011). A high throughput h.264/AVC intra-frame encoding loop architecture for hd1080p. In: *2011 IEEE International Symposium of Circuits and Systems (ISCAS)* (pp. 579–582). https://doi.org/10.1109/ISCAS.2011.5937631
11. Hase, T., Hintermaier, W., Frey, A., Strobel, T., Baumgarten, U., & Steinbach, E. (2011). Influence of image/video compression on night vision based pedestrian detection in an automotive application. In: *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)* (pp. 1–5).
12. Hsiang, H., Chen, K., Li, P., & Chen, Y. (2020). Analysis of the effect of automotive ethernet camera image quality on object detection models. In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* (pp. 021–026).
13. Information technology–generic coding of moving pictures and associated audio information–part 1: Systems. Standard, International Organization for Standardization, Geneva, CH (2019).
14. Series H: Audiovisual multimedia systems, Infrastructure of audiovisual services - Coding of moving video, Advanced video coding for generic audiovisual services. Standard, International Telecommunication Union, Geneva, CH (2012).
15. Jung, J. S., Jo, Y. J., & Lee, H. J. (2011). A fast h.264 intra frame encoder with serialized execution of 4×4 and 16×16 predictions and early termination. *Journal of Signal Processing Systems, 64*(1), 161–175. https://doi.org/10.1007/s11265-010-0574-6
16. Kaijin, W., Zhang, S., Jia, H., Xie, D., & Gao, W. (2012). A flexible and high-performance hardware video encoder architecture. https://doi.org/10.1109/PCS.2012.6213368
17. Keshaveni, N., Ramachandran, S., & Gurumurthy, K. S. (2009). Design and implementation of integer transform and quantization processor for h.264 encoder on FPGA. In: *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies* (pp. 646–649). https://doi.org/10.1109/ACT.2009.164
18. Kthiri, M., Kadionik, P., Lévi, H., Loukil, H., Ben Atitallah, A., & Masmoudi, N. (2010). An FPGA implementation of motion estimation algorithm for h.264/AVC. In: *2010 5th International Symposium On I/V Communications and Mobile Network* (pp. 1–4). https://doi.org/10.1109/ISVC.2010.5654826
19. Kuo, H., Wu, L., Huang, H., Hsu, S., Lin, Y. (2011). A low-power high-performance h.264/AVC intra-frame encoder for 1080phd video. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 19*(6), 925–938. https://doi.org/10.1109/TVLSI.2010.2045402
20. Lee, Y. G., & Song, B. C. (2009). An intra-frame rate control algorithm for ultralow delay h.264/advanced video coding (AVC). *IEEE Transactions on Circuits and Systems for Video Technology, 19*(5), 747–752. https://doi.org/10.1109/TCSVT.2009.2017413
21. Lin, Y., Ku, C., Li, D., & Chang, T. (2009). A 140-mhz 94 k gates hd1080p 30-frames/s intra-only profile h.264 encoder. *IEEE Transactions on Circuits and Systems for Video Technology, 19*(3), 432–436. https://doi.org/10.1109/TCSVT.2009.2013511
22. Lin, Y., Li, D., Lin, C., Kuo, T., Wu, S., Tai, W., Chang, W., & Chang, T. (2008). A 242mw 10mm2 1080p h.264/AVC high-profile encoder chip. In: *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers* (pp. 314–615). https://doi.org/10.1109/ISSCC.2008.4523183
23. Lin, Y-L. S., Kao, C-Y., Kuo, H-C., Chen, & J-W. (2010). *VLSI Design for Video Coding: H.264/AVC Encoding from Standard Specification to Chip*. Springer US. https://doi.org/10.1007/978-1-4419-0959-6
24. Liu, Z., Song, Y., Shao, M., Li, S., Li, L., Ishiwata, S., Nakagawa, M., Goto, S., & Ikenaga, T. (2009). HDTV1080p h.264/AVC encoder chip design and performance analysis. *IEEE Journal of Solid-State Circuits 44*(2), 594–608. https://doi.org/10.1109/JSSC.2008.2010797
25. Matsui, H., Ogawa, T., Mochizuki, A., Nakayama, H., Kodama, S., Moriya, A., Koto, S., & Ishiwata, S. (2011). An h.264 full HD 60i double speed encoder IP supporting both MBAFF and field-pic structure. In: *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test (pp. 1–4)*. https://doi.org/10.1109/VDAT.2011.5783632

26. Mukherjee, R., Chakrabarti, I., & Sengupta, S. (2012). FPGA based architectural implementation of context-based adaptive variable length coding (CALVC) for h.264/AVC. In: *IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012)* (pp. 1–4). https://doi.org/10.1049/cp.2012.2417
27. Mukherjee, R., Keyur, S., Sandeep, E., Chakrabarti, I., & Sengupta, S. (2013). FPGA based implementation of quantization and its inverse for h.264 codec. In: *2013 IEEE Conference on Information Communication Technologies* (pp. 986–989). https://doi.org/10.1109/CICT.2013.6558240
28. Pastuszak, G. (2014). FPGA architectures of the quantization and the dequantization for video encoders. In: *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems* (pp. 290–293). https://doi.org/10.1109/DDECS.2014.6868812
29. Pastuszak, G. (2015). Architecture design of the h.264/AVC encoder based on rate-distortion optimization. *IEEE Transactions on Circuits and Systems for Video Technology, 25*(11), 1844–1856. https://doi.org/10.1109/TCSVT.2015.2402911
30. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91
31. Reeba, K., & Perinbam, J. (2008). FPGA implementation of integer transform and quantizer for h.264 encoder. *Signal Processing Systems, 53*, 261–269. https://doi.org/10.1007/s11265-008-0163-0
32. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., & Chen, L. (2018). *Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation*. CoRR abs/1801.04381. http://arxiv.org/abs/1801.04381
33. Siblini, A., Baaklini, E., Sbeity, H., Fadlallah, A., & Niar, S. (2013). Efficient FPGA implementation of h.264 CACLV entropy decoder. In: *2013 8th IEEE Design and Test Symposium* (pp. 1–3). https://doi.org/10.1109/IDT.2013.6727116
34. Song, B. C., Yi, Y., Lee, Y. G., Kim, N. H., Ko, J. H., Kim, T. H., Lim, D. K., Ju, W. H., Moon, J. P., & Cho, K. (2012). 1080p 60 hz intra-frame video codec chip design and its implementation. *Journal of Signal Processing Systems, 67*(3), 291–303. https://doi.org/10.1007/s11265-010-0564-8
35. Stabernack, B., Möller, J., Hahlbeck, J., & Brandenburg, J. (2015). Demonstrating an FPGA implementation of a full HD real-time HEVC decoder with memory optimizations for range extensions support. In: *2015 Conference on Design and Architectures for Signal and Image Processing (DASIP)* (pp. 1–2). https://doi.org/10.1109/DASIP.2015.7367247
36. Stabernack, B., & Steinert, F. (2021). Architecture of a low latency h.264/AVC video codec for robust ml based image classification. In: *Workshop on Design and Architectures for Signal and Image Processing, DASIP '21* (14th ed., pp. 1–9). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3441110.3441149
37. Steinert, F., Schelten, N., Schulte, A., & Stabernack, B. (2020) Hardware and software components towards the integration of network-attached accelerators into data centers. In: *2020 23rd Euromicro Conference on Digital System Design (DSD)* (pp. 149–153). https://doi.org/10.1109/DSD51259.2020.00033
38. Chen, T-C., Chien, S-Y., Huang, Y-W., Tsai, C-H., Chen, C-Y., Chen, T-W., & Chen, L-G. (2006). Analysis and architecture design of an HDTV720p 30 frames/s h.264/AVC encoder. *IEEE Transactions on Circuits and Systems for Video Technology, 16*(6), 673–688. https://doi.org/10.1109/TCSVT.2006.873163
39. Wei, L., Ding, D.d., Du, J., Yu, B. B., & Yu, L. (2011). An efficient hardware design for HDTV h.264/AVC encoder. *Journal of Zhejiang University Science C, 12*(6), 499–506. https://doi.org/10.1631/jzus.C1000201
40. Wu, J., Yuen, C., Cheung, N., Chen, J., & Chen, C. W. (2015). Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications. *IEEE Transactions on Circuits and Systems for Video Technology, 25*(12), 1988–2001. https://doi.org/10.1109/TCSVT.2015.2441412
41. Zheng, J., Xu, C., & Guo, J. (2012). An effective approach for hardware design of intra prediction in h.264/AVC. In: *Anti-counterfeiting, Security, and Identification* (pp. 1–4). https://doi.org/10.1109/ICASID.2012.6325313

Fritjof Steinert received the Bachelor of Engineering in Electronics and Communication Systems in 2010 and the Master of Engineering in Communication and Information Technology in 2011 from the University of Applied Sciences Beuth-Hochschule für Technik Berlin, Germany. In 2012, he joined the Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute in Berlin as research associate, currently in the Embedded Systems group in the Video Communications and Applications department. Since 2021 he is a PhD student at the chair Embedded Systems Architectures for Signal Processing at the University of Potsdam. His current research interests include the use of hardware accelerator technologies mainly based on network technologies in data centers or embedded environments and their design, especially for image processing.

Benno Stabernack received the Diploma and Dr.-Ing. degrees in electrical engineering from the Technical University of Berlin, Germany in 1996 and 2004, respectively. In 1996 he joined the Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute (HHI), Berlin, Germany, where as the head of the Embedded Systems Group of the Video Communication and Applications department, he is currently responsible for research projects focused on hardware and software architectures for image and video processing algorithms. Since summer 2005 he is giving a lectures on the design of application specific processors at the Technical University of Berlin. Since October 2016 he holds the chair of "Embedded Systems Architectures for Signal Processing" at the University of Potsdam as a joint appointment with the Fraunhofer Institute for Telecommunications–Heinrich-Hertz-Institute (HHI), Berlin, Germany. His current research interests include VLSI architectures for video signal processing, machine learning, application specific processor architectures for embedded media signal processing and System-on-Chip (SOC) design. He has been engaged in several national, international and European research projects as well in the standardization process of ITU H.266 video coding standard.