Computer Science • 19(3) 2018 https://doi.org/10.7494/csci.2018.19.3.2825

Maciej Wielgosz
Michał Karwatowski
Marcin Pietroń
Kazimierz Wiatr

FPGA IMPLEMENTATION OF PROCEDURES FOR VIDEO QUALITY ASSESSMENT

Abstract *The video resolutions used in a variety of media are constantly rising. While manufacturers struggle to perfect their screens, it is also important to ensure the high quality of the displayed image. Overall quality can be measured using a Mean Opinion Score (MOS). Video quality can be affected by miscellaneous artifacts appearing at every stage of video creation and transmission. In this paper, we present a solution to calculate four distinct video quality metrics that can be applied to a real-time video quality assessment system. Our assessment module is capable of processing 8K resolution in real time set at a level of 30 frames per second. The throughput of 2.19 GB/s surpasses the performance of pure software solutions. The module was created using a high-level language to concentrate on architectural optimization.*

Keywords video quality, video metrics, image processing, FPGA, Impulse C

Citation Computer Science 19(3) 2018: 279–305

1. Introduction

In addition to traditional Quality of Service (QoS), Quality of Experience (QoE) nowadays poses a real challenge for Internet audiovisual service providers, broadcasters, and new Over-The-Top (OTT) services. The churn effect is linked to the impact of QoE; end-user satisfaction is a real added value in this competition. However, QoE tools should be proactive and innovative solutions that are well-adapted to new audiovisual technologies. Therefore, objective audiovisual metrics are frequently dedicated to monitoring, troubleshooting, investigating, and setting benchmarks of content applications working in real time or offline.

The so-called Full-Reference (FR), Reduced-Reference (RR), and No-Reference (NR) quality metrics are used for models standardized according to International Telecommunication Union -Telecommunication Standardization Sector (ITU-T) recommendations. Most of the models have some limitations, as they were usually validated using one of the following hypotheses [30]:

- frame freezes last for up to two seconds;
- there is no degradation at the beginning or end of a video sequence;
- there are no skipped frames;
- video reference is clean (no spatial or temporal distortion);
- there is a minimum delay supported between video reference and the video (sometimes with a constant delay);
- up- or down-scaling operations are not always taken into account.

In the past, metrics based on three historical video artifacts (blockiness, jerkiness, and blur) were sufficient for providing an efficient predictive result. Consequently, most models are based on measuring these artifacts for producing a predictive Mean Opinion Score (Mean Opinion Score (MOS)). In other words, the majority of the algorithms generating the predicted MOS show a mix of blur, blockiness, and jerkiness metrics. The weighting between each of these Key Performance Indicators (KPIs) could be a simple mathematical function. If one of the KPIs is not correct, the global predictive score is completely wrong. Other KPIs are usually not taken into account (exposure time, distortion, interlacing, etc.) in predicting MOS [30].

ITU-T has been working on KPI-like distortions for many years (please refer to [12] for more information). The history of the recommendations is shown in Table 1, while metrics based only on the video signal are shown in Table 2, both based on [30]. Related research in [9] addresses the measurement of multimedia quality in mobile networks with an objective parametric model [30].

ITU-T Study Group 12 (SG12) is currently working on modeling standards for multimedia and Internet Protocol Television (IPTV) based on bit-stream information. The Q14/12 work group is responsible for the projects provisionally known as non-intrusive parametric model for assessment of performance of multimedia streaming (P.NAMS) and non-intrusive bit-stream model for assessment of performance of multimedia streaming (P.NBAMS) [30].

Table 1
History regarding ITU-T Recommendations (based on [30])

Model Type	Format	Rec.	Year
FR	SD	J.144 [14]	2004
FR	QCIF–VGA	J.247 [18]	2008
RR	QCIF–VGA	J.246 [17]	2008
FR	SD	J.144 [14]	2004
RR	SD	J.249 [21]	2010
FR	HD	J.341 [22]	2011
RR	HD	J.342 [23]	2011
Bitstream	VGA–HD	In progress	Exp. 2014
Hybrid	VGA–HD	In progress	Exp. 2014

Table 2
Synthesis of FR, RR, and NR MOS models (based on [30, 51])

		FR	RR	NR
5*Resolution	HDTV	J.341 [22]	n/a	n/a
	SDTV	J.144 [14]	n/a	n/a
	VGA	J.247 [18]	J.246 [17]	n/a
	CIF	J.247 [18]	J.246 [17]	n/a
	QCIF	J.247 [18]	J.246 [17]	n/a

P.NAMS utilizes packet-header information (e.g., from IP through MPEG2-TS), while P.NBAMS also uses payload information (i.e., coded bit-stream) [45]. However, this work focuses on the overall quality (in MOS units), while monitoring of audiovisual quality by key indicators (MOAVI) is focused on KPIs [30].

Most of the recommended models are based on a global quality evaluation of video sequences (as can be found in the P.NAMS and P.NBAMS projects). The predictive score is correlated with subjective scores obtained with global evaluation methodologies such as SAMVIQ, DSCQS, and ACR. The duration of video sequences is limited to 10 or 15 seconds to avoid the forgiveness effect (the observer is unable to score the video properly after 30 seconds and may give more weight to artifacts occurring at the end of a sequence). When one model is deployed for monitoring video services, the global scores are provided for fixed temporal windows and without any acknowledgment of the previous scores [30].

The time needed to process such metrics is long, even when a powerful machine is used. Hence, the measurement periods have been short and never extended. As a result, the measurements miss sporadic and erratic audiovisual artifacts.

The concept proposed here (partly based on the framework for the integrated video quality assessment published in [34]) can isolate and focus investigations, set up algorithms, increase monitoring periods, and guarantee better predictions. Depending on the technologies used in audiovisual services, the impact of QoE can change

completely. The scores are separated for each algorithm and preselected before the testing phase. Then, each KPI can be analyzed by working on the spatially and temporally perceived axes. The standard metric cannot provide appropriate predictive scores with certain new audiovisual artifacts such as exposure distortions. Moreover, it is important to detect the artifacts as well as the experience described and detected by consumers. In real-life situations, customers can call a helpline when the video quality of their audiovisual services decreases and describe the annoyance and visibility problems; they are not required to provide a MOS.

There are many possible reasons for video disturbance, and they can arise at any point along the video chain transmission (filming stage to end-user stage). The main concern of the authors of the papers is the efficient hardware implementation of the proposed solution. This problem is addressed using hardware-development techniques that decrease the latency and throughput of the system (which is a challenging task). The following papers partially cover the issue of developing hardware accelerators for various tasks: [25, 27, 48–50].

The main contribution of this work is a solution capable of the simultaneous real-time calculation of four distinct video quality metrics for video streams with resolutions of up to 8K.

This paper is organized as follows. Section 2 provides information about existing works on Video Quality Assessment. Section 3 presents the quality metrics used in this paper. Section 4 described the tools needed for high-level FPGA programming. Section 5 presents the hardware platform utilized in our experiments. Section 6 provides a detailed description of the system implementation in Impulse C. Finally, section 7 contains the results of the experiment. Section 8 summarizes the contribution of this work.

2. Related work

Automated video quality assessment has been an issue addressed in many papers in recent years. [31] presented a no-reference solution for a MPEG video stream that measured quantization error and the blocking effect. Their solution showed a positive correlation with other methods. However, because of the technology available at the time of publication, their system throughput was far from modern requirements. [36] successfully implemented the Levenberg-Marquardt method in low-end platforms using VHDL. They showed that the hardware-implementation results maintain a strong correlation with software solutions despite the reduced precision due to the usage of fixed-point arithmetic. [35] implemented field-offset detection as well as blurring and ringing measurements in Field-Programmable Gate Array (FPGA). Their language of choice was Verilog; they achieved real-time processing for full HD resolutions using a platform based on Virtex 4. In [46], the authors presented a survey of video quality metrics and found full-reference methods reliable; however, they pointed out that they are computationally intensive. They also noted that a combination of no-reference methods also covers a broad range of operational conditions.

Paper [29] describes a machine-learning approach combining the results of video quality metrics; for no-reference, they obtained a correlation ranging from 0.85 to 0.9 (with the subjective score provided by human observers). Active development of such tools as [33] and [47] (which are used by major media companies) show just how important the role played by automated video quality metrics is.

3. Video quality assessment

This paper addresses the challenging task of building a module capable of accelerating the computations of the metrics. Consequently, the designed module produces video quality assessment in real time for each video frame. The following four metrics were implemented in the hardware:

- blockiness,
- exposure,
- blackout,
- interlace.

The choices of metrics were driven by their performance and hardware implementation feasibility. The authors designed and implemented a single module for each of the four metrics. Such an approach enables the hardware units to share among the metrics architectures, and it boosts the overall throughput of the video assessment quality module. The blockiness and exposure metrics are presented in [42, 43], respectively.

This section presents an overview of all of the metrics and the algorithms used in this work. The notation used in equations is presented in Table 3.

Table 3
Notation used in equations

Symbol	Description
BLX	number of horizontal blocks in a frame
BLY	number of vertical blocks in a frame
$sortMeanBL$	ordered sequence of the average luminance of blocks
$sortSumBL$	ordered sequence of the luminance sums calculated for each block

3.1. Blocking

Blocking is caused by the independence of the calculations for each block in the image. While many compression algorithms divide frames into blocks, this is one of the most popular and visible artifacts. Because of the coarse quantization, the correlation among the blocks is lost, and horizontal and vertical borders appear. Another reason might be a change in resolution when a small picture is scaled up to be displayed on a larger screen.

The blockiness metric used in this work is based on [5]. This metric assumes a constant block size (which was chosen to be 8×8 pixels). The metric value depends

on two factors: magnitude of the color difference at the block's boundary and the picture contrast near the boundaries (see Fig. 1).

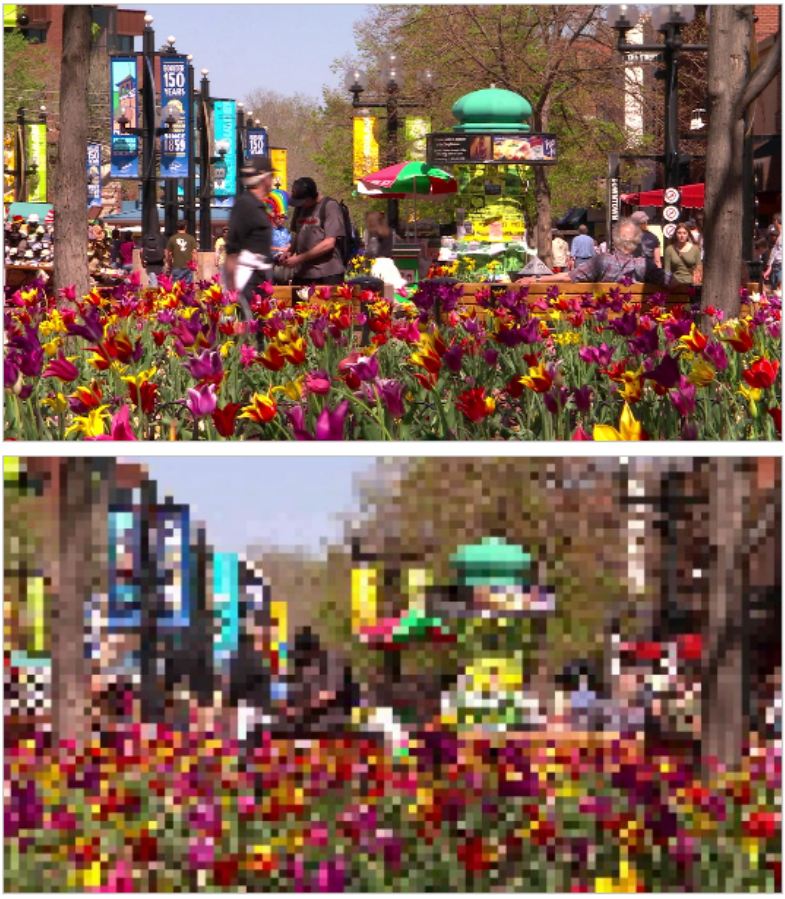

Figure 1. Blockiness artifact

Consequently, the *InterSum* and *IntraSum* values are computed for each incoming frame.

1. *InterSum* is the sum of the absolute differences between the pixels located on the border of two neighboring picture blocks – Equation (1).
2. *IntraSum* is the sum of the absolute differences between the pixels located directly next to the neighboring pixel of the picture block – Equation (2).

$$
\begin{aligned}
&\mathit{InterSum}_{x,y} = \\
&\quad |b_{x,y}(29) - b_{x+1,y}(25)| + |b_{x,y}(30) - b_{x+1,y}(26)| + |b_{x,y}(31) - b_{x+1,y}(27)| + \\
&\quad |b_{x,y}(32) - b_{x+1,y}(28)| + |b_{x,y}(61) - b_{x+1,y}(57)| + |b_{x,y}(62) - b_{x+1,y}(58)| + \\
&\quad |b_{x,y}(36) - b_{x,y+1}(35)| + |b_{x,y}(40) - b_{x,y+1}(39)| + |b_{x,y}(44) - b_{x,y+1}(43)| + \\
&\quad |b_{x,y}(48) - b_{x,y+1}(47)| + |b_{x,y}(52) - b_{x,y+1}(51)| + |b_{x,y}(56) - b_{x,y+1}(55)| \,. \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{IntraSum}_{x,y} = \\
&\quad |b_{x,y}(29) - b_{x+1,y}(1)| + |b_{x,y}(30) - b_{x+1,y}(2)| + |b_{x,y}(31) - b_{x+1,y}(3)| + \\
&\quad |b_{x,y}(32) - b_{x+1,y}(4)| + |b_{x,y}(61) - b_{x+1,y}(33)| + |b_{x,y}(62) - b_{x+1,y}(34)| + \\
&\quad |b_{x,y}(36) - b_{x,y+1}(1)| + |b_{x,y}(40) - b_{x,y+1}(5)| + |b_{x,y}(44) - b_{x,y+1}(9)| + \\
&\quad |b_{x,y}(48) - b_{x,y+1}(13)| + |b_{x,y}(52) - b_{x,y+1}(17)| + |b_{x,y}(56) - b_{x,y+1}(21)| \,. \qquad (2)
\end{aligned}
$$

The computing schemes of *InterSum* and *IntraSum* are depicted in Figure 2, along with the pixel numeration scheme. $b_{x,y}(i)$ used in Equations (1) and (2) means the i-th pixel of the x, y block.

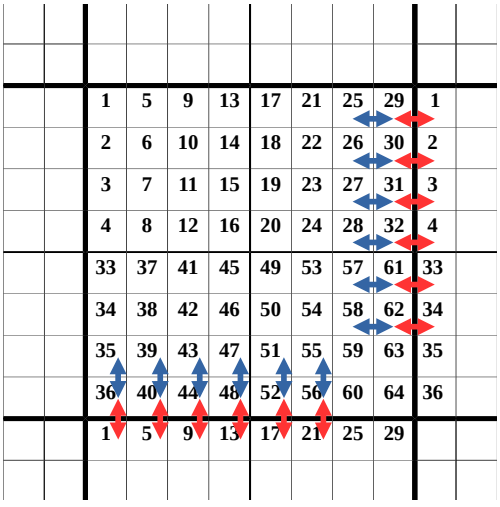

Figure 2. Model of video-coding block with pixel-numeration scheme: blue – pixels used to calculate *IntraSum*; red – pixels used to calculate *InterSum*

The blockiness metric is the ratio of *IntraSum* to *InterSum*, as presented by Equation (3).

$$
\mathit{blockinessMetric} = \frac{\sum_{y=2}^{BLY-1} \sum_{x=2}^{BLX-1} \mathit{IntraSum}_{x,y}}{\sum_{y=2}^{BLY-1} \sum_{x=2}^{BLX-1} \mathit{InterSum}_{x,y}}. \qquad (3)
$$

3.2. Exposure time distortions

Exposure time distortions are visible as imbalances in the brightness (frames that are too dark or too bright). These are caused by incorrect exposure times or recording video without sufficient lighting. It is also possible to cause this distortion by improper digital enhancement. Histograms of various exposure levels for the same image are presented in Figure 3.

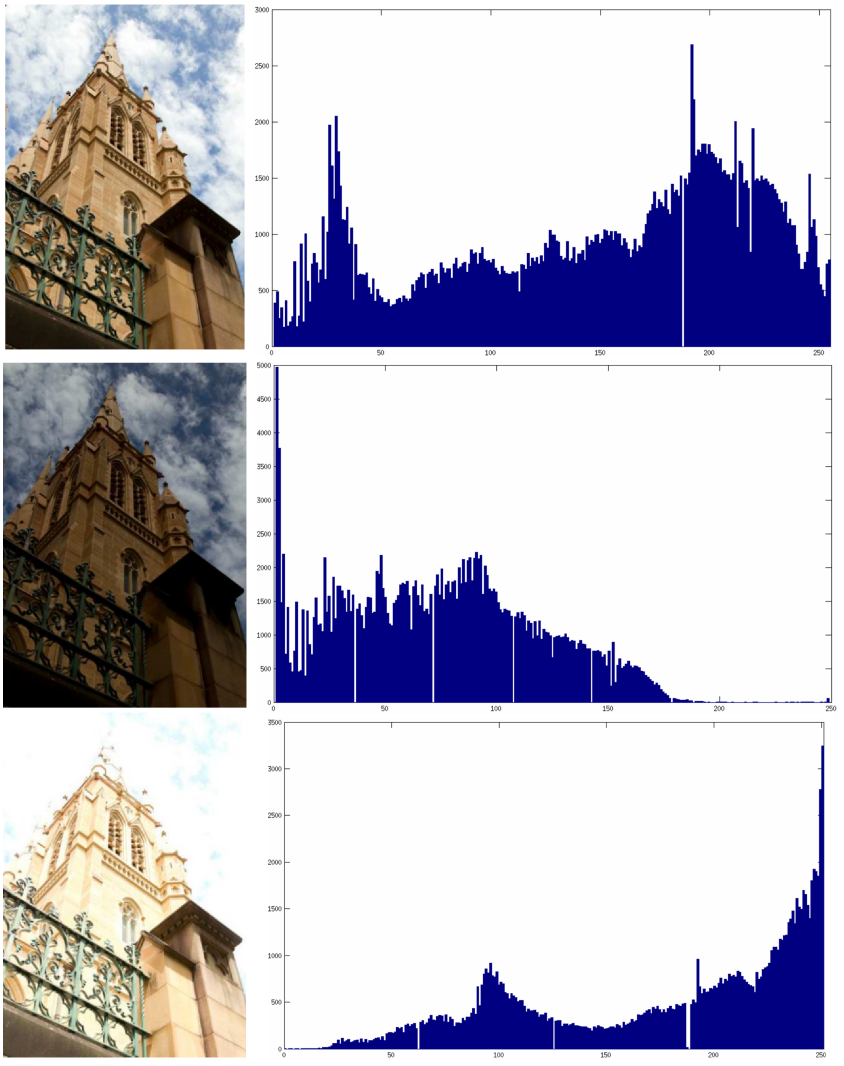

Figure 3. Luminance histograms of correct (top), underexposed (middle), and overexposed (bottom) images

The mean brightness of the darkest and brightest parts of the image is calculated in order to detect this distortion. The exposure metric is presented in Equation (4), where L_d, Equation (5) represents the three darkest blocks and L_b, Equation (6) represents the three brightest blocks. The number 3 was chosen arbitrarily; however, it may prove inefficient for higher resolutions and would need to be increased.

$$exposureMetric = \frac{L_b + L_d}{2}. \tag{4}$$

$$L_d = \sum_{i=1}^{3} sortMeanBL_i. \tag{5}$$

$$L_b = \sum_{i=BLX\times BLY-2}^{BLX\times BLY} sortMeanBL_i. \tag{6}$$

The results of the metrics mentioned above were mapped to the Mean Opinion Score (MOS). The thresholds referred to the MOS scale, which determine the score below in which each distortion is noticeable.

3.3. Blackout

This is manifested as the picture disappearing – a black screen. It appears when all packets of data are lost or as a result of incorrect video recording. Image blackout detection is independent of the frame color; i.e., the detection result is positive (it equals '1') if the frame has a uniform color; otherwise, the result is '0.' A comparison of all of the pixels of the frame under consideration seems to be the most straightforward approach.

Unfortunately, this is a very computationally demanding method that requires n comparisons, where n is the number of pixels within the frame. The authors came up with an alternative method that utilizes partial results of the exposure time distortion method. This resulted in a significant reduction in the metric implementation cost.

The new metric description:

A frame is split into blocks of 8×8 pixels. The sum of the luminance is calculated for each block. If the difference between the block of the highest luminance and the lowest is lower than the *thBlout* threshold, the detection result equals '1'; otherwise it is '0' (*thBlout* is set to a constant four).

$$blackoutMetric = \begin{cases} 0 \text{ if } sortSumBL_{BLX\times BLY} - sortSumBL_1 \geqslant thBlout \\ \\ 1 \text{ otherwise} \end{cases}. \tag{7}$$

3.4. Interlace

Interlace is a technique where a single frame is a composition of two half-frames, each of which contains half of the information. The odd half-frames contain the odd rows of pixels, while the even half-frames contain the even rows of pixels. The resulting frame is created by interlacing both of them. The idea of interlace is presented in Figure 4. Interlace distortion becomes visible when the two half-frames are not properly aligned. It is especially visible for videos that include motion.

Figure 4. Creation of interlaced frame

The authors have proposed their solution for the interlace distortion metric. It is calculated independently for each micro 4×4 pixel block and then subsequently combined into a complete metric. A given block is marked as a block with interlace distortion if a change of luminance of Row 1 relative to Row 2 is in the same direction for all pixel pairs (as presented in Figure 5). The change between the pixels in Row 2 and Row 3 is in the opposite direction, and finally, the change between Rows 3 and 4 is in the same direction as for 1 and 2.

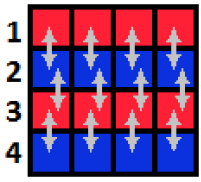

Figure 5. Interlaced microblock model

Equation (8) determines if a given block has interlace distortion, where $d_{i,j}$ is the j-th difference between the luminance values of the i-th micro block. Equation (9) calculates the metric value for the whole frame.

$$interlace_i = \begin{cases} 1 \text{ if } \sum_{j=1}^{12} |sgn\,(d_{i,j})| = 12 \\ 0 \text{ otherwise} \end{cases}. \tag{8}$$

$$interlaceMetric = \frac{\sum_{i=1}^{4 \times BLX \times BLY} intelace_i}{4 \times BLX \times BLY}. \tag{9}$$

Figure 6 illustrates the detection of an interlace in a sample frame. The effect is the most visible in shapes containing sharp vertical lines. The presented solution shows positive results.

Figure 6. Detection of interlace artifact – gray pixels indicate distortion detection

4. High-level hardware design – tools and methodology

The module was implemented using the Impulse C language. Impulse C is a high-level language based on a Stream-C compiler (which was created at the Los Alamos National Laboratories in the 1990s). The idea evolved into a corporation named Impulse Accelerated Technologies Company (2002), which is now a supporting vendor of Impulse C and the holder of the Impulse C rights. The primary intention of the language designers was to bridge the gap between the hardware and software and facilitate the process of system-level design. It was achieved through abstracting most of the language constructs so that designers can focus on the algorithm rather than low-level details of the implementation [8].

There is a whole set of high-level languages such as Dime-C, SystemC, Handel-C, and Mitrion C available nowadays that enable the specification and implementation of the system at the module level. However, most of them introduce their structures (e.g., Mitrion C), expanding or modifying the existing standards of high-level languages. On the one hand, such an approach helps to establish a design space by imposing a strict language expression set. On the other hand, designers have to comprehend a whole range of language structures along with their appropriate application schemes (which can be pretty tedious).

Such an extra effort is justified in the case of people who expect to use the tool for a reasonably long time (professional digital logic designers). Unfortunately, most of the FPGA High Level Language (HLL) users are people familiar with programming languages (e.g., C, C++, Java, Fortran) who need to port some part of their application into hardware. Therefore, it seems reasonable to leverage one of the well-known standards such as ANSI C. Moreover, ANSI C allows for access to the low-level details of an application, which is very useful in some cases. It can be said that *C gives the lowest possible level of abstraction among the high-level languages*. The ideas mentioned above prevailed in the design of the Impulse C language.

There are several features of the Impulse C language that, in the authors' view, are superior to other currently used HLLs. First of all, Impulse C allows designers to reuse their HDL code by providing mechanisms that facilitate the incorporation of existing modules. Furthermore, three different architectures are supported: *combinational*, *pipelined*, and *asynchronous*, which cover a complete range of existing design scenarios. Secondly, C compatibility makes it easy for software engineers to switch from General Purpose Processor (GPP) programming to FPGA design. It also provides a platform for software-hardware integration within one design environment. Finally, Impulse C comes with a range of Platform Support Packages (PSPs) that provide a communication interface between the FPGA and GPP computational nodes. Furthermore, PSPs usage provides for the portability of an application across different platforms. In fact, PSPs are packs of files that describe a system's profile to the Impulse C compiler [3]. The compiler uses this information to generate the interface components needed to connect the hardware processes to a system bus and interconnect them inside the FPGA and also establish the software side of any software/hardware connections such as stream, signal, and memory [3, 37, 38].

The language enables both fine-grain and coarse-grain parallelism; the former is implemented within a process, whereas the latter is built from multiple-process structures. It is worth noting that algorithm partitioning must be handled by a programmer – this stage is not automated by the compiler, which means that it is up to the designer to classify the different sections of an application. However, due to the portability of the code, it is possible to migrate between the hardware and software sections if adequate language structures are employed. Concerning this, it is recommended that one avoids using language constructs that confine a given part of the code to the software or hardware solely.

A designer should keep the number of control signals and branches low, since the primary goal of the HLL FPGA algorithm implementation is to increase throughput at the expense of latency (trading latency for throughput). Using control signals may compromise this effort and should make a designer rethink the concept of the architecture.

The Impulse C compiler automatically generates test benches, software-hardware interfaces, and synthesizable HDL code; it automatically finds parallel structures in the code as well. However, a good coding practice is the explicit indication of the sections that should be parallelized. Both the hardware and software parts of the code can be compiled with GNU Compiler Collection (GCC).

Impulse C can be characterized as a stream-oriented process-based language. The processes are the main building blocks interconnected using streams to form an architecture for the desired hardware module. From the hardware perspective, the processes and streams are hardware modules and First In, First Out (FIFO) registers, respectively. The Impulse C programming model is based on the *Communicating Sequential Processes* model [38] and is illustrated in Figure 7. Each process must be classified as a hardware or software process.

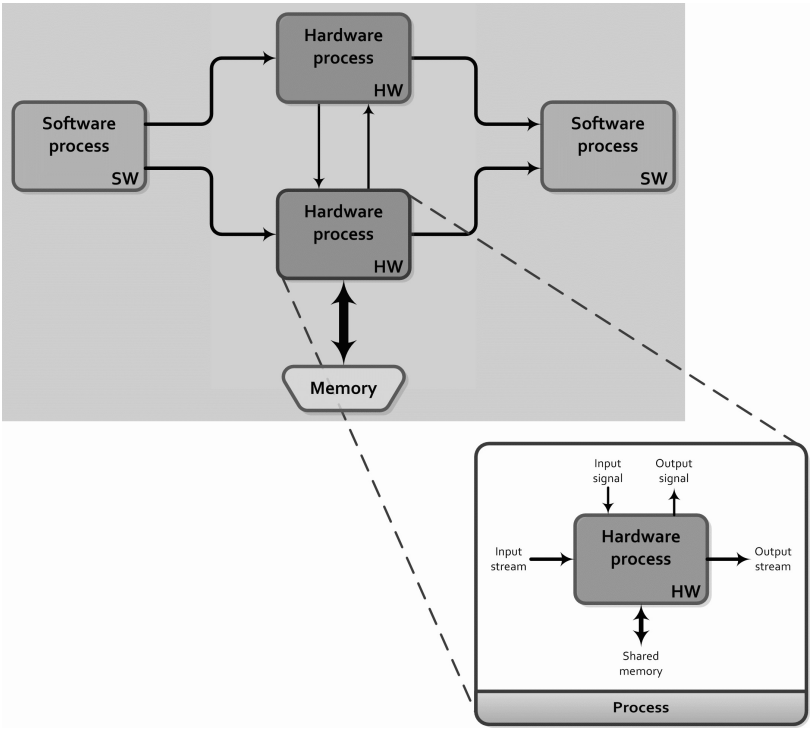

Figure 7. Impulse C programming model

It is the programmer's responsibility to ensure the interprocess synchronization. Like most of the HLLs, Impulse C does not provide access to the clock signal, which relieves the designer from implementing cycle synchronization procedures. However, it is possible to attach HDL modules and synchronize them at the level of the RTL using the clock signal.

5. FPGA-based platform

The module was implemented on the Pico M503 platform [2] connected through PCIe to a server with an Intel i7-950 processor and 12 GB of RAM. The Pico platform (Fig. 8) consists of two components:

- EX-500 board with a Gen2 PCI-Express controller, which enables the connection of up to six FPGAs circuits to the motherboard;
- M503 FPGA boards [2].

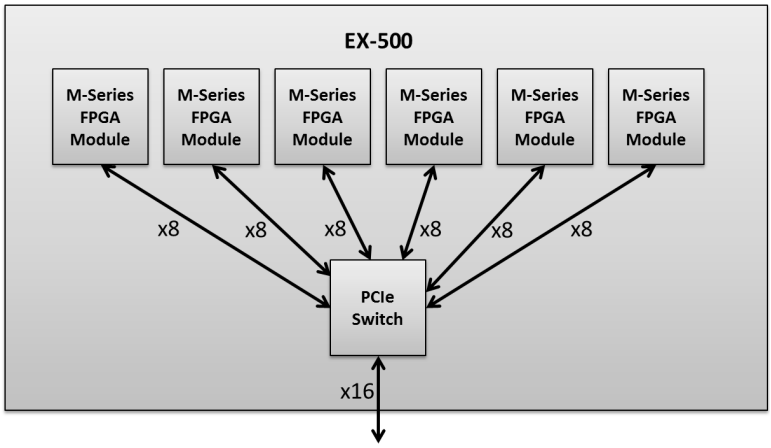

Figure 8. FPGA-based platform used for computations – Pico Computing

Communication between a CPU and the FPGA is realized with eight lines of PCIe interface – full-duplex connection streams. If more than two boards are used, the throughput is limited to 5 GB/s; in the case of using only one board, the maximum throughput reaches about 3 GB/s. Another limitation is the width of the stream, which is equal to 128 bits.

6. Impulse C implementation of module

This section describes the hardware implementation of the video quality module. Subsection 6.1 shows a general concept of the module; next, the description is divided into two parts according to the two parts of the projects in the Impulse C language: software (6.2) and hardware (6.3).

6.1. Architecture of module

The block diagram of the video quality assessment module is shown in Figure 9. It consists of three subblocks:

- producer – reads video data from a file and sends it to the vqFPGA block using the InputStream;
- vqFPGA – reads data from the InputStream, executes the video quality metrics, and sends the results to the Consumer process using the OutputStream;
- consumer – reads data from the OutputStream, analyzes it, and sends it to the standard output stream.

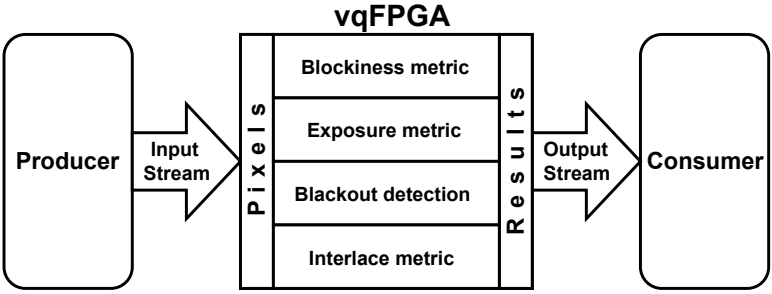

Figure 9. Architecture of video quality assessment module as implemented in Impulse C

The width of the Input and the Output stream is 128 bits, which is the maximum width of the Pico M503 platform stream. The scheme described above is parallelized sixfold, there are six producers, vqFPGA circuits and consumer processes in the real module.

Each Impulse C project is composed of a software and hardware part, as is the video quality assessment module.

6.2. Software part of module

The software part is composed of three functions: producer, consumer, and the main program function (which is used to launch the FPGA-based accelerator and all of the application-related threads). It is also responsible for programming the FPGA with a bit file. The producer function opens the input stream to the FPGA and sends pre-read video data. The Pico module input stream is 128-bits wide; thus, it is recommended that one organizes the data in such chunks so that the best possible throughput is achieved. Each 8×8 block is divided into four microblocks. Each microblock contains 16 values, eight bits each, representing pixels in gray scale. Such a structure allows for sending the whole block in four-bus clock cycles, retaining data consistency. The described scheme is presented in Figure 10 on an example of a 32×32 image; the same approach was used for larger images.

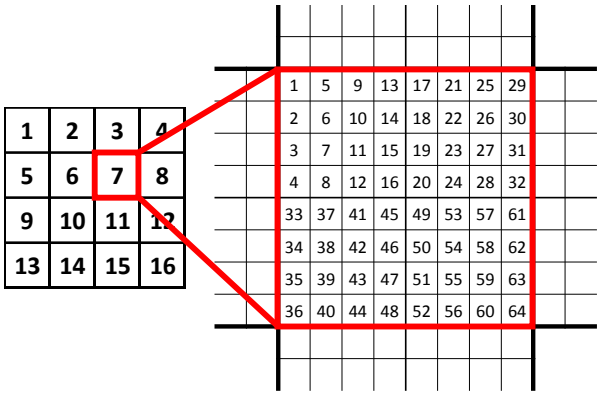

Figure 10. Structure of sample block and corresponding transfer sequence of sample frame (left) and order of pixels within each block (right)

The consumer function manages the module output stream. At the end of each video frame, a valid-results frame is received. Its size is also fixed to 128-bits wide, as it fits best to the hardware. A special structure of the results frame was designed as presented in Figure 11. The frame contains the results of the calculation of the blackout, exposure, and interlace distortion metrics. The last part of computing the blockiness metric is performed in the software; thus, each frame contains the required values of *InterSum* and *IntraSum*. Finally, distinct metrics can be combined into one score (this, however, is not a part of this work).

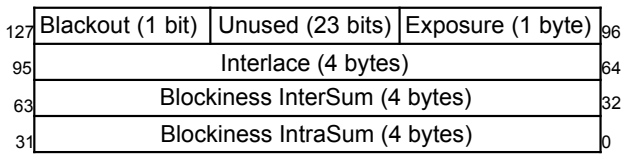

Figure 11. Structure of frame sent through OutputStream

6.3. Hardware part of module

The hardware part is composed of vqFPGA modules and the additional logic that handles the data-fetching and -sending results to the software part. The hardware part is equipped with two data streams corresponding to the software streams, which are opened before the data transfer is conducted and closed once it is finished. The hardware module requires information about the video resolution to be sent in advance to the actual stream. Each 128-bit word is then arranged into a microblock. Afterwards, the data is sent to the parts of the hardware responsible for computing each metric.

The module registers are reset after all of the microblocks of a given frame are processed and a new frame comes in. The maximum number of combinational stages

between registers were experimentally determined as 64 and implemented with a `Co Set stageDelay` Impulse C pragma. This also requires the use of a `Co Pipeline` pragma, which implements the pipelined design approach.

6.3.1. Blockiness metric

For the blockiness metric, only the most computationally demanding parts were implemented in the hardware. The *InterSum* and *IntraSum* are calculated inside the FPGA, while the final division is done in the software. As presented in Figure 2, the calculations require data from the neighboring blocks, and storing all necessary data inside the FPGA would be very inconvenient. Therefore, the authors modified the data-sending scheme to make it more suitable for calculating the blockiness metric. The first row and first column are omitted, and the block boundaries are shifted as presented in Figure 12.

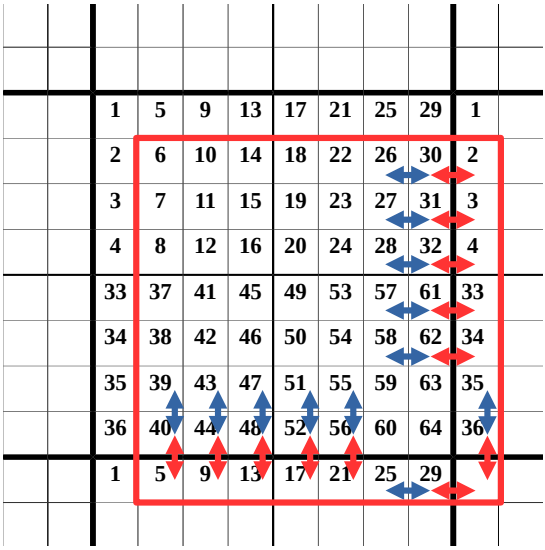

Figure 12. Shifted block

After such operations, all of the data necessary for the *InterSum* and *IntraSum* calculations is available in a single block. Any influence of this operation on the other metrics was not observed.

The source code presented in Figure 13 shows the hardware implementation of the blockiness metric. Due to the efficient data serialization, the module is implemented with few lines of code (which also results in low hardware resource consumption). It is worth noting that the source code reflects the operations described in Equation (3).

```
if(microBlock % 4 == 2){
  IntraSum += ABSDIFF(p9,p5) + ABSDIFF(p10,p6) +
                      ABSDIFF(p11,p7) + ABSDIFF(p12,p8);
  InterSum += ABSDIFF(p9,p13) + ABSDIFF(p10,p14) +
                      ABSDIFF(p11,p15) + ABSDIFF(p12,p16);
}
if(microBlock % 4 == 3){
  IntraSum += ABSDIFF(p2,p3) + ABSDIFF(p6,p7) +
                      ABSDIFF(p10,p11) + ABSDIFF(p14,p15);
  InterSum += ABSDIFF(p4,p3) + ABSDIFF(p8,p7) +
                      ABSDIFF(p12,p11) + ABSDIFF(p16,p15);
}
if(microBlock % 4 == 0){
  IntraSum += ABSDIFF(p9,p5) + ABSDIFF(p8,p12) +
                      ABSDIFF(p2,p3) + ABSDIFF(p14,p15);
  InterSum += ABSDIFF(p9,p13) + ABSDIFF(p12,p16) +
                      ABSDIFF(p4,p3) + ABSDIFF(p15,p16);
}
```

Figure 13. Blockiness metric source code

6.3.2. Exposure time distortions metric

The metric is composed of three steps. In the first one, a luminance mean value of each code block is calculated. Then, six extreme values for each frame are found (the three smallest and three largest). The extreme values are used to compute the mean value.

Several modifications were introduced to adapt it to the hardware implementation. The size of each block is constant; therefore, instead of the mean, the sum of the values may be used. This removes the division operation in a mean calculation, which is very resource-demanding for block sizes that are not to the power of two. The fractional part may be disregarded, as it holds little importance. Without changing the algorithm, the mean may be computed for eight results (the four largest and four smallest), which enables the use of a bit shift operation (shift right by two bits) instead of the very expensive division.

The sum of the luminance values is stored in a *blockSum* variable. The extreme blocks are searched for (Fig. 14), and the sum of their luminance values are stored in variables *blockSumMAX1–4* and *blockSumMIN1–4*.

The result of the metric is a weighted mean of the luminance of the pixels from the extreme blocks. All of the *blockSumMAX* and *blockSumMIN* parameters are summed up, and the result is shifted left by nine bits (because $2^9 = 512 = 8 * 64$; 8 is the number of extreme blocks, and 64 is the number of pixels within a single block). To prevent data range overflow (`co_uint16` is used), each datum is shifted right by two bits; the result is subsequently moved by the remaining seven bits. Variables

microBlock (which is used to select each micro block) and *blockSumMAX* are reset after all of the data results are sent to the software part of the module. *blockSumMIN* is set to 16.384 before the next frame is taken from the input.

```
if(blockSum < blockSumMIN4){
    if (blockSum < blockSumMIN3){
        if(blockSum < blockSumMIN2){
            if(blockSum < blockSumMIN1){
                blockSumMIN4 = blockSumMIN3;
                blockSumMIN3 = blockSumMIN2;
                blockSumMIN2 = blockSumMIN1;
                blockSumMIN1 = blockSum;
            }
            else {
                blockSumMIN4 = blockSumMIN3;
                blockSumMIN3 = blockSumMIN2;
                blockSumMIN2 = blockSum;
            }
        }
        else {
            blockSumMIN4 = blockSumMIN3;
            blockSumMIN3 = blockSum;
        }
    }
    else
        blockSumMIN4 = blockSum;
}
```

Figure 14. Implementation of exposure metric – minimal luminance values of frame

6.3.3. Blackout metric

The blackout metric is implemented as four lines of Impulse C code (Fig. 15). The module is comprised of one adder/subtractor and one comparator. The metric result is sent to the software part of the module as a single bit set to '1' in the OutputStream, which indicates that blackout occurred.

```
if((blockSumMAX1 - blockSumMIN1) > thBlout)
    blackoutMetric = 0;
else
    blackoutMetric = 1;
```

Figure 15. Implementation of blackout metric

6.3.4. Interlace distortion metric

The way the data is structured and transferred between the hardware and software parts is presented in 6.2. It is adapted to this particular metric and improves the performance of the module. A single microblock is sent, and the interlace distortion detection is conducted just by examining the *IS_INTERLACE* and *IS_INTERLACE2* (Fig. 16) conditions.

```
#define IS_INTERLACE ((p1>p2) && (p5>p6) && (p9>p10)\
            && (p13>p14) && (p3>p2) && (p7>p6) && (p11>p10)\
            && (p15>p14) && (p3>p4) && (p7>p8) && (p11>p12)\
            && (p15>p16))

#define IS_INTERLACE2 ((p1<p2) && (p5<p6) && (p9<p10)\
            && (p13<p14) && (p3<p2) && (p7<p6) && (p11<p10)\
            && (p15<p14) && (p3<p4) && (p7<p8) && (p11<p12)\
            && (p15<p16))
```

Figure 16. Implementation of interlace distortion metric

If one of these conditions is met, the result of the metric is incremented by one. The sum of all of the microblocks of a frame is a maximum possible value of the result that affected the choice of the variable used to store it; i.e., `co_uint32`. After all of the microblocks of the frame are received, the variable is reset. The module is composed of 12 interconnected comparators that form a single huge XNOR gate. In addition, the module is comprised of an adder and 32-bit shift register for the *interlaceMetric* variable.

7. Experimental results

Several experiments were conducted to determine the performance of the module. Figure 17 presents the performance of both the hardware and software implementation of the video quality assessment module for a variety of resolutions. This started from the very low resolutions of QVGA (320×240) and VGA (640×480) through full HD (1920×1080) to UHD resolutions of 4K (4096×2160) and 8K (7680×4320). Due to the variety of display aspect ratios, we chose the power of two values to determine the aspect ratios for 4K and 8K.

The resulting image was around 1 % larger than the popular 16 : 9. The green line indicates the real-time processing performance (assuming that the video is streamed at the 30-frames-per-second rate). The hardware version of the video quality assessment module is capable of processing 8K in real-time. Table 4 presents the hardware resource consumption of registers (#reg) and lookup tables (#lut) in the Pico platform as a function of the number of vqFPGA modules implemented as well as the corresponding throughput achieved.

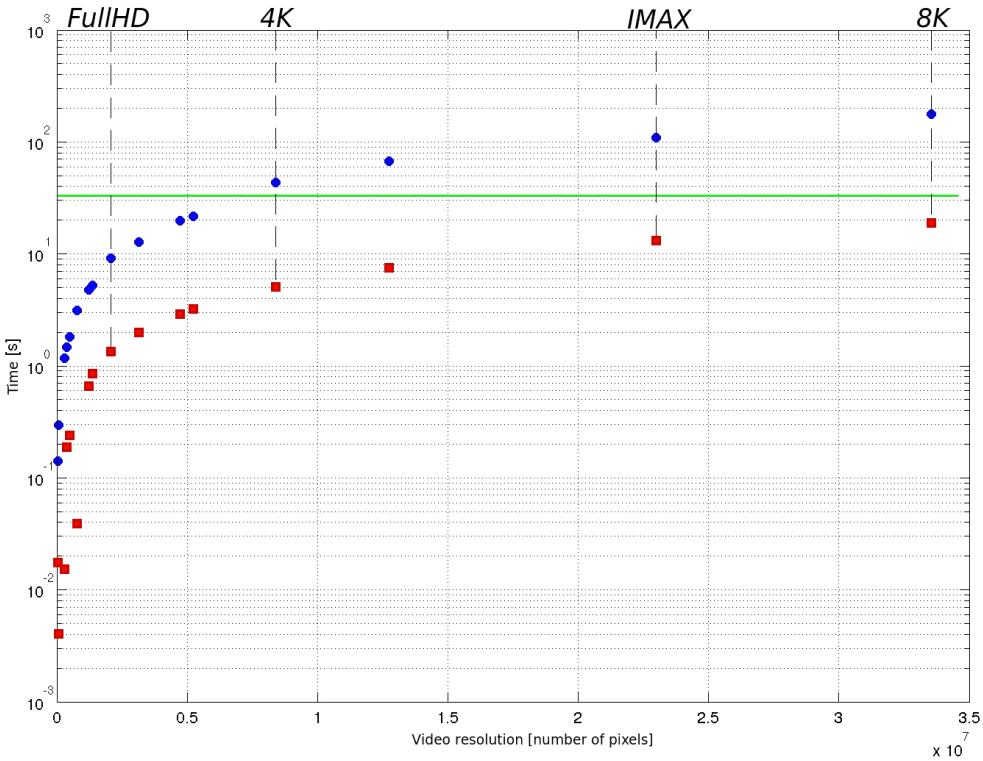

Figure 17. Processing time of 1000 frames as function of video resolution for both hardware (red squares) and software (blue dots) implementations

Table 4
Hardware resource consumption (Xilinx Virtex-6 LX240T FPGA)

# vqFPGA modules	#reg	#lut	throughput [GB/s]
1	23 982 (7 %)	11 836 (6 %)	0.66
2	29 179 (10 %)	15 915 (7 %)	1.27
3	34 182 (12 %)	19 330 (9 %)	1.6
4	39 379 (15 %)	23 214 (11 %)	1.96
5	44 576 (18 %)	27 398 (13 %)	2.11
6	49 773 (20 %)	31 320 (14 %)	2.19

Figure 18 presents the acceleration results as a function of video resolution. It is worth noting that the resolution has a direct impact on the size of a single chunk of data sent over InputStream, which in turn affects the transfer rate and overall processing time. The acceleration (Fig. 18) is a speed-up achieved by the hardware solution as compared to the software one.

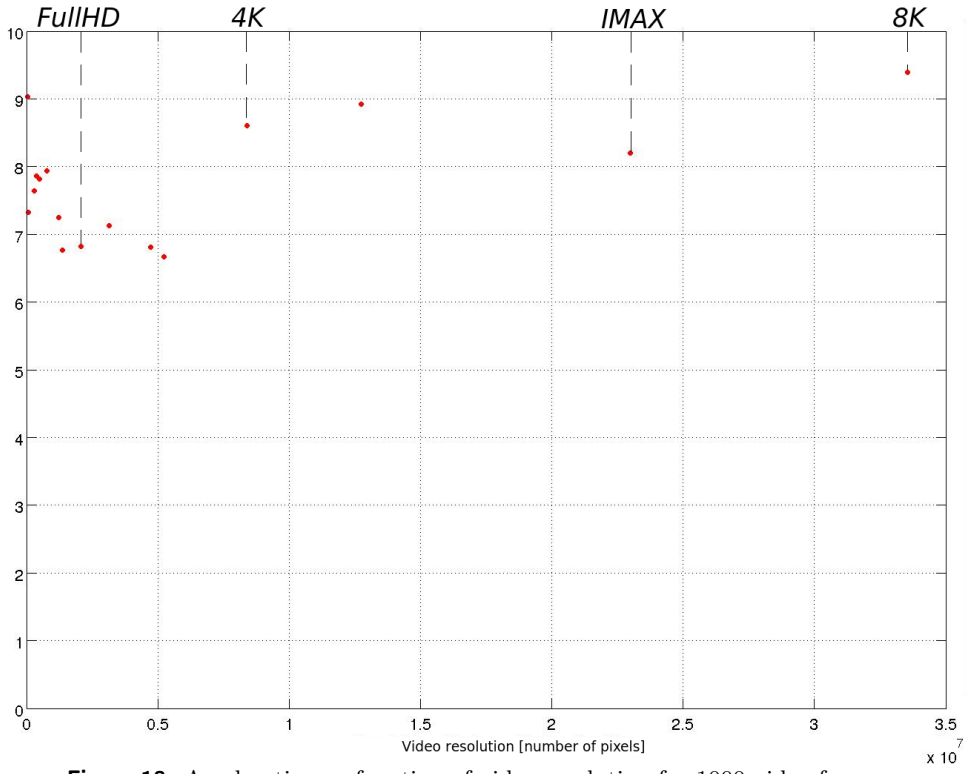

Figure 18. Acceleration as function of video resolution for 1000 video frames

8. Summary

The presented solution is capable of simultaneously calculating four distinct video quality assessment metrics on a single video stream. The hardware platform allowed for the real-time processing of 8K resolution. The solution based on the CPU only did not meet the real-time requirements for resolutions higher than full HD. The whole project was implemented using Impulse C, a high-level language that significantly reduced design time, facilitated the system integration process, and enabled architectural optimization, which boosted the overall performance of the solution. Some improvements can still be made – more metrics can be added, and their combination should be compared to a subjective score. Also, due to the low resource utilization, more parallel modules can be implemented inside the FPGA, which could further speed up the calculations. If a theoretical highest throughput for the Pico M503 platform (3 GB/s) will be reached, it would allow for the processing of 16K resolution with 24 fps, which is the minimum that can be considered as a real time. However, because the Impulse C language allows for the seamless movement of the design between different platforms, the presented solution can utilize more efficient hardware with Platform Support Package provided to achieve even better results.

Acknowledgements

This work was performed thanks to the funds for AGH statutory activity 11.11.230.017

References

[1] CENELEC EN 50132, Alarm systems. CCTV surveillance systems for use in security applications, 2011.

[2] Pico computing M503 manual, 2013. http://picocomputing.com/products/hpc-modules/m-503/. Accessed: 23.06.2016.

[3] Bodenner R.: Creating Platform Support Packages, 2011. http://www.impulseaccelerated.com/AppNotes/APP109_PSP/IATAPP109_PSP.pdf. Accessed: 11.06.2018.

[4] Cerqueira E., Janowski L., Leszczuk M., Papir Z., Romaniak P.: Video Artifacts Assessment for Live Mobile Streaming Applications. In: Mauthe A., Zeadally S., Cerqueira E., Curado M. (eds.), *Future Multimedia Networking*, *Lecture Notes in Computer Science*, vol. 5630, pp. 242–247. Springer Berlin Heidelberg, 2009. http://dx.doi.org/10.1007/978-3-642-02472-6_26.

[5] Farias M.C., Mitra S.K.: No-reference video quality metric based on artifact measurements. In: *IEEE International Conference on Image Processing 2005*, vol. 3, pp. III–141, 2005. http://dx.doi.org/10.1109/ICIP.2005.1530348.

[6] Farias M.C.Q., Heynderickx I., Macchiavello Espinoza B.L., Redi J.A.: Visual Artifacts Interference Understanding and Modeling (VARIUM). In: *Seventh International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, vol. 1. Scottsdale, 2013.

[7] Farias M.C.Q., Mitra S.K.: Perceptual contributions of blocky, blurry, noisy, and ringing synthetic artifacts to overall annoyance, *Journal of Electronic Imaging*, vol. 21(4), pp. 043013–043013, 2012. http://dx.doi.org/10.1117/1.JEI.21.4.043013.

[8] Gancarczyk G., Wielgosz M., Wiatr K.: An introduction to offloading CPUs to FPGAs – Hardware programming for software developers, 2013. http://www.eetimes.com/document.asp?doc_id=1280560. Accessed: 11.06.2018.

[9] Gustafsson J., Heikkila G., Pettersson M.: Measuring multimedia quality in mobile networks with an objective parametric model. In: *15th IEEE International Conference on Image Processing, ICIP 2008*, pp. 405–408, 2008. http://dx.doi.org/10.1109/ICIP.2008.4711777.

[10] International Telecommunication Union: ITU-T P.800, Methods for subjective determination of transmission quality, 1996. http://www.itu.int/rec/T-REC-P.800-199608-I.

[11] ITU: ITU-T P.830, Subjective performance assessment of telephone-band and wideband digital codecs, 1996. http://www.itu.int/rec/T-REC-P.830-199602-I/en.

[12] ITU: ITU-T P.930, Principles of a reference impairment system for video, 1996. http://www.itu.int/rec/T-REC-P.930-199608-I.

[13] ITU: ITU-T P.910, Subjective video quality assessment methods for multimedia applications, 1999. http://www.itu.int/rec/T-REC-P.910-200804-I.

[14] ITU: ITU-T J.144, Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference, 2004. http://www.itu.int/rec/T-REC-J.144-200403-I.

[15] ITU: ITU-R BT.1702, Guidance for the reduction of photosensitive epileptic seizures caused by television, 2005. http://www.itu.int/rec/R-REC-BT.1702.

[16] ITU: ITU-T G.100/P.10, Vocabulary for performance and quality of service, 2006. http://www.itu.int/rec/T-REC-P.10-200607-I.

[17] ITU: ITU-T J.246, Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference, 2008. http://www.itu.int/rec/T-REC-J.246-200808-I.

[18] ITU: ITU-T J.247, Objective perceptual multimedia video quality measurement in the presence of a full reference, 2008. http://www.itu.int/rec/T-REC-J.247-200808-I.

[19] ITU: ITU-T P.912, Subjective video quality assessment methods for recognition tasks, 2008. http://www.itu.int/rec/T-REC-P.912-200808-I.

[20] ITU: ITU-R BT.500-12, Methodology for the subjective assessment of the quality of television pictures, 2009. http://www.itu.int/rec/R-REC-BT.500-12-200909-I.

[21] ITU: ITU-T J.249, Perceptual video quality measurement techniques for digital cable television in the presence of a reduced reference, 2010. http://www.itu.int/rec/T-REC-J.249-201001-I.

[22] ITU: ITU-T J.341, Objective perceptual multimedia video quality measurement of HDTV for digital cable television in the presence of a full reference, 2011. http://www.itu.int/rec/T-REC-J.341-201101-I.

[23] ITU: ITU-T J.342, Objective multimedia video quality measurement of HDTV for digital cable television in the presence of a reduced reference signal, 2011. http://www.itu.int/rec/T-REC-J.342-201104-I.

[24] ITU: ITU-T H.264, Advanced video coding for generic audiovisual services, 2012. http://www.itu.int/rec/T-REC-H.264-201201-P/en.

[25] Jamro E., Wielgosz M., Wiatr K.: FPGA Implementation of Strongly Parallel Histogram Equalization. In: Girard P., Krasniewski A., Gramatová E., Pawlak A., Garbolino T. (eds.), *DDECS*, pp. 93–98. IEEE Computer Society, 2007. http://dx.doi.org/10.1109/DDECS.2007.4295260.

[26] Karunasekera S.A., Kingsbury N.G.: A distortion measure for blocking artifacts in images based on human visual sensitivity, *IEEE Transactions on Image Processing*, vol. 4(6), pp. 713–724, 1995.

[27] Karwatowski M., Russek P., Wielgosz M., Koryciak S., Wiatr K.: Energy Efficient Calculations of Text Similarity Measure on FPGA-Accelerated Computing Platforms. In: Wyrzykowski R., Deelman E., Dongarra J., Karczewski K., Kitowski J., Wiatr K. (eds.), *PPAM (1)*, *Lecture Notes in Computer Science*, vol. 9573, pp. 31–40, Springer, 2015. https://doi.org/10.1007/978-3-319-32149-3_4.

[28] van Kester S., Xiao T., Kooij R.E., Brunnström K., Ahmed O.K.: Estimating the impact of single and multiple freezes on video quality. In: *Proceedings SPIE, Human Vision and Electronic Imaging XVI*, vol. 7865, pp. 78650O–78650O–10. 2011. http://dx.doi.org/10.1117/12.873390.

[29] Le Callet P., Viard-Gaudin C., Péchard S., Caillault É.: No reference and reduced reference video quality metrics for end to end QoS monitoring, *IEICE transactions on communications*, vol. 89(2), pp. 289–296, 2006.

[30] Leszczuk M., Hanusiak M., Blanco I., Dziech A., Derkacz J., Wyckens E., Borer S.: Key indicators for monitoring of audiovisual quality. In: Cetin E.A., Salerno E. (eds.), *MUSCLE International Workshop on Computational Intelligence for Multimedia Understanding*, p. 6. ERCIM, IEEE Xplore, Antalya, Turkey, 2013.

[31] Lu L., Wang Z., Bovik A.C., Kouloheris J.: Full-reference video quality assessment considering structural distortion and no-reference quality evaluation of MPEG video. In: *Proceedings of IEEE International Conference on Multimedia and Expo, ICME'02*, vol. 1, pp. 61–64, 2002. http://dx.doi.org/10.1109/ICME.2002.1035718.

[32] Moore G.W.: Flatterland. Like Flatland. Only More So.: I. Stewart; Perseusblishing Cambridge, MA, 2001, pp 301, *Neurocomputing*, vol. 42(1-4), pp. 337–338, 2002. http://dx.doi.org/10.1016/S0925-2312(01)00637-3.

[33] MSU Video Quality Measurement tools. http://www.compression.ru/video/quality_measure/. Accessed: 11.06.2018.

[34] Mu M., Romaniak P., Mauthe A., Leszczuk M., Janowski L., Cerqueira E.: Framework for the integrated video quality assessment. In: *Multimedia Tools and Applications*, vol. 61(3), pp. 787–817, 2012. http://dx.doi.org/10.1007/s11042-011-0946-3.

[35] Neborovski E., Marinkovic V., Katona M.: Video quality assessment approach with field programmable gate arrays. In: *The 33rd International Convention MIPRO*, pp. 713–717, 2010.

[36] de Oliveira M., da Silva W., Fonseca K.V.O., Pohl A.D.A.P.: VHDL implementation of a No-Reference video quality metric using the Levenberg-Marquardt method. In: *International Symposium on Broadband Multimedia Systems and Broadcasting*, pp. 1–5, 2014. http://dx.doi.org/10.1109/BMSB.2014.6873474.

[37] Papu J.J., See O.H.: Design of a reconfigurable computing platform. In: *Proceedings of Innovative Technologies in Intelligent Systems and Industrial Applications*, pp. 148–153, 2009.

[38] Pellerin D., Thibault S.: *Practical FPGA programming in C*, Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, 2005.

[39] Punchihewa A., Bailey D.G.: Artefacts in image and video systems: classification and mitigation. In: *Proceedings of Image and Vision Computing New Zealand 2002*, pp. 197–202, 2002.

[40] Rohaly A.M., Corriveau P., Libert J., et al.: Video Quality Experts Group: Current Results and Future Directions, 2000.

[41] Romaniak P.: *Assessment of perceptual video quality affected by acquisition and bit-rate reduction artifacts*, Ph.D. thesis, 2011.

[42] Romaniak P., Janowski L., Leszczuk M., Papir Z.: A no reference metric for the quality assessment of videos affected by exposure distortion. In: *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2011. http://dx.doi.org/10.1109/ICME.2011.6011903.

[43] Romaniak P., Janowski L., Leszczuk M., Papir Z.: Perceptual quality assessment for H.264/AVC compression. In: *Consumer Communications and Networking Conference (CCNC)*, pp. 597–602, 2012. http://dx.doi.org/10.1109/CCNC.2012.6181021.

[44] Takahashi A., Schmidmer C., Lee C., Speranza F., Okamoto J., Brunnström K., Janowski L., Barkowsky M., Pinson M., Staelens N., Huynh Thu Q., Green R., Bitto R., Renaud R., Borer S., Kawano T., Baroncini V., Dhondt Y.: *Report on the Validation of Video Quality Models for High Definition Video Content*, Tech. rep., Video Quality Experts Group, 2010.

[45] Takahashi A., Yamagishi K., Kawaguti G.: Global Standardization Activities Recent Activities of QoS/QoE Standardization in ITU-T SG12, *Ntt Technical Review*, vol. 6(9), pp. 1–5, 2008.

[46] Torres Vega M., Sguazzo V., Mocanu D.C., Liotta A.: An experimental survey of no-reference video quality assessment methods, *International Journal of Pervasive Computing and Communications*, vol. 12(1), pp. 66–86, 2016. http://dx.doi.org/10.1108/IJPCC-01-2016-0008.

[47] Video Multi-Method Assessment Fusion. https://github.com/Netflix/vmaf. Accessed: 11.06.2018.

[48] Wielgosz M., Panggabean M., Chilwan A., Rønningen L.A.: FPGA-Based Platform for Real-Time Internet. In: Stoica A., Zarzhitsky D., Howells G., Frowd C.D., McDonald-Maier K.D., Erdogan A.T., Arslan T. (eds.), *Proceedings of Third International Conference on Emerging Security Technologies, Lisbon, 2012*, IEEE Computer Society, pp. 131–134, 2012. http://dx.doi.org/10.1109/EST.2012.18.

[49] Wielgosz M., Panggabean M., Rønningen L.A.: FPGA Architecture for Kriging Image Interpolation, *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 4(12), 2013. http://ijacsa.thesai.org/.

[50] Wielgosz M., Panggabean M., Wang J., Rønningen L.A.: An FPGA-Based Platform for a Network Architecture with Delay Guarantee, *Journal of Circuits, Systems, and Computers*, vol. 22(6), 2013. http://dx.doi.org/10.1142/S021812661350045X.

[51] Wyckens E.: Proposal studies on new video metrics. In: Webster A. (ed.), *VQEG Hillsboro Meeting*, p. 17. Orange Labs, Video Quality Experts Group (VQEG), Hillsboro, 2011.

Affiliations

Maciej Wielgosz
AGH University of Science and Technology, Academic Computer Center CYFRONET, Krakow, Poland, wielgosz@agh.edu.pl, ORCID ID: https://orcid.org/0000-0002-4401-2957

Michał Karwatowski
AGH University of Science and Technology, Academic Computer Center CYFRONET, Krakow, Poland, mkarwat@agh.edu.pl, ORCID ID: https://orcid.org/0000-0001-6285-136X

Marcin Pietroń
AGH University of Science and Technology, Academic Computer Center CYFRONET, Krakow, Poland, pietron@agh.edu.pl, ORCID ID: https://orcid.org/0000-0001-9357-9231

Kazimierz Wiatr
AGH University of Science and Technology, Academic Computer Center CYFRONET, Krakow, Poland, wiatr@agh.edu.pl, ORCID ID: https://orcid.org/0000-0001-5959-0277

Received: 22.03.2018
Revised: 19.06.2018
Accepted: 20.06.2018