CAST White Paper

Video Streaming with Near-Zero Latency Using Altera Arria V FPGAs and Video and Image Processing Suite Plus the Right Encoder

Nikos Zervas, CAST, Inc. — January 2016

In the video world, *latency* is the amount of time between the instant a frame is captured and the instant that frame is displayed. *Low latency* is a design goal for any system where there is real-time interaction with the video content, such as video conferencing or drone piloting.

But the meaning of low latency can vary, and the methods for achieving low latency aren't always obvious.

Here we'll define and explain the basics of video latency, and discuss how one of the biggest impacts in reducing latency comes from choosing the right video encoding.

Using this understanding and drawing from cores in the Altera® Video and Image Processing Suite, developers of delay- and power-sensitive systems can readily deliver competitive products using Arria® V FPGAs that meet or exceed the most challenging specifications. Moreover, the quality and easy reuse of the MegaCore® and other intellectual property (IP) cores, their consistent use of a common interface, and the productivity of Altera's tools and design kits mean developers can deliver these systems on time and on budget.

Characterizing Video System Latency

There are several stages of processing required to make the pixels captured by a camera visible on a video display. The delays contributed by each of these processing steps—as well as the time required for transmitting the compressed video stream—together produce the total delay, which is sometimes called *end-to-end latency*.

Measuring Video Latency

Latency is colloquially expressed in time units, for example seconds (s) or milliseconds (ms).

The biggest contributors to video latency are the processing stages that require temporal storage of data, for example short-term *buffering* in some form of memory. Because of this, video system engineers tend to measure latency in terms of the buffered video data, for example, a latency of two frames or eight horizontal lines.

Contacts: +1 201.391.8300 • info@cast-inc.com • www.cast-inc.com

Converting from frames to time depends on the video's frame rate. For example, a delay of one frame in 30 frames-per-second (fps) video corresponds to 1/30th of a second (33.3 ms) of latency.

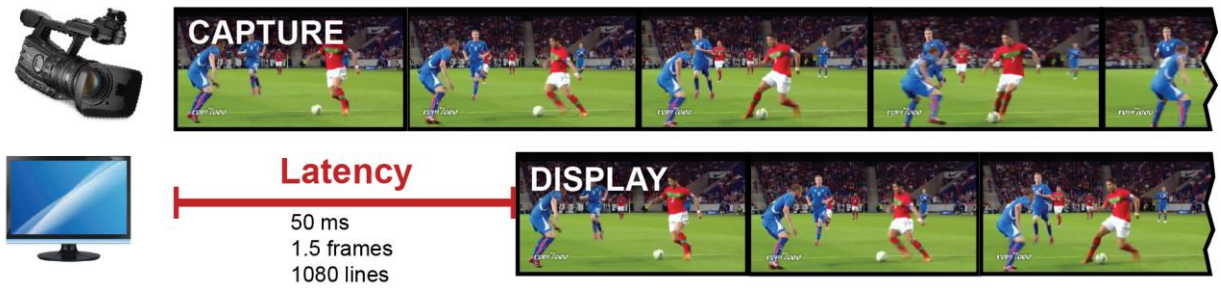

Figure 1: Representing latency in a 1080p30 video stream.

Converting from video lines to time requires both the frame rate and the frame size or resolution. A 720p HD video frame has 720 horizontal lines, so a latency of one line at 30 fps is 1/(30*720) = 0.046 ms of latency. In 1080p @ 30 fps, that same one-line latency takes a much briefer 0.030 ms.

Defining Low Latency

There is no universal absolute value that defines *low latency*. Instead, what is considered acceptable low latency varies by application.

When humans interact with video in a live video conference or when playing a game, latency lower than 100 ms is considered to be low, because most humans don't perceive a delay that small. But in an application where a machine interacts with video—as is common in many automotive, industrial, and medical systems—then latency requirements can be much lower: 30 ms, 10 ms, or even under a millisecond, depending on the requirements of the system.

You will also see the term *ultra-low latency* applied to video processing functions and IP cores. This is a marketing description and not a technical definition, which means "really, really low latency" for the given application.

Designing for Low Latency in a Video-Streaming Application

Because it is commonplace in today's connected, visual world, let's examine latency in systems that stream video from a camera (or server) to a display over a network.

As with most system design goals, achieving suitably low latency for a streaming system requires tradeoffs, and success comes in achieving the optimum balance of hardware, processing speed, transmission speed, and video quality. As previously mentioned, any temporary storage of uncompressed or compressed video data increases latency, so reducing buffering is a good primary goal.

Video data buffering is imposed whenever processing must wait until some specific amount of data is available. The amount of data buffering required can vary from a few pixels, to several video lines, or even to a number of whole frames. With a target maximum acceptable latency in mind, we can easily calculate the amount of data

buffering the system can tolerate, and hence to what level—pixel, line, or frame—one should focus on when budgeting and optimizing for latency.

For example, with our human viewer's requirement of 100 ms maximum latency for a streaming system using 1080p30 video, we can calculate the maximum allowable buffering through the processing pipeline as follows:

100 ms/(33.3 ms per frame) = 3 frames, or
1,080 lines per frame x 3 frames =3,240 lines, or
1,920 pixels per line x 3,240 lines = 6.2 million pixels

In this context, we can see that worrying about the latency of a hardware JPEG encoder—typically just a few thousand pixels—is irrelevant because it is too small to make any significant difference in end-to-end latency. Instead, one should focus on the points of the system where entire frames or large number of video lines are buffered.

Representative results from such a focused design effort are itemized in Table 1, which provides the distribution of latency from the various stages of a carefully designed low-latency video-streaming system. Here all unnecessary frame-level buffering has been eliminated and hardware codecs have been used throughout because software codecs typically feature higher latencies due to latency overheads related to memory transfers and task-level management from the operating system (OS).

Processing Stage	Buffering	Latency (1080p30)
Capture post-processing (e.g., Bayer filter, chroma resampling)	A few lines (e.g. 8)	< 0.50 ms
Video compression (e.g. Motion-JPEG, MPEG-1/2/4 or H.264 with single-pass bitrate regulation)	16 lines for conversion from raster scan A few thousand pixels on the encoder pipeline	0.49 ms ~0.01 ms
Bit-Rate Averaging Buffer (BRAB)	From a number of frames (e.g. more than 30) to deep sub-frame (e.g. 1/4 frame)	from 1 s to 8.33 ms
Network processing (e.g. RTP/UDP/IP encapsulation)	A few kilobytes (KB)	<0.01 ms
Decoder Stream Buffer (DSB)	From a number of frames (e.g. more than 30) to sub-frame (e.g. 1/4 frame)	from 1 s to 8.33 ms
Video decompression (JPEG, MPEG-1/2/4, or H.264)	16 lines for conversion from raster scan A few thousand of pixels on the decoder pipeline	0.49ms ~0.01 ms
Display pre-processing (e.g. Scaling, Chroma Resampling)	A few lines (e.g. 8)	< 0.50 ms
Display controller buffer	From one frame in most cases to a few lines or tens of lines (e.g. 64)	from 33.3 ms to 2 ms

Table 1. Latency Sources in a Low-Latency, 1080p30 Video-Streaming System

As in most video-streaming applications, the dominant latency contributor is the DSB and the BRAB. Next, we look at what this is, why we need those, and how we can best reduce the latency they introduce.

DSB and BRAB, the Dominant Latency Contributors

In the Table 1 example, the DSB and BRAB may add from 1 second down to 8.33 milliseconds of latency each. This large range depends on the video stream's bit rate attributes. What attributes can we control to keep the DSB and BRAB delay on the lower end of this range?

The Illusion of Constant Bit Rate

The bandwidth limitations of a streaming video system usually require regulation of the transmission bit rate. For example, a 720p30 video might need to be compressed for successful transmission over a channel that has a bit rate limited to 10 megabits per second (Mbps).

One could reasonably assume that bit rate regulation yields a transmission bit rate that is constant at every point in time, for example every frame travels at the same 10 Mbps. But this turns out not to be true, and that is why we need the DSB to buffer the stream for the receiving decoder and the BRAB to buffer the stream before we send it over a bandwidth-limited channel. Let's look closer at how this bit rate regulation works in video compression.

Video compression reduces video data size by using fewer bits to represent the same video content. However, not all types of video content are equally receptive to compression. In a given frame, for example, the flat background parts of the image can be represented with much fewer bits than are necessary for the more detailed foreground parts. In a similar way, high-motion sequences need many more bits than do those with moderate or no motion.

As a result, compression natively produces streams of *variable bit rate* (VBR). With bit rate regulation (or bit-rate control), we force compression to produce the same amount of stream data over equal periods of time (for example, for every 10 frames, or each 3 second interval). We call this *constant bit rate* (CBR) video. It comes at the expense of video quality, as we are in effect asking the compression engine to assign bits to content based on time rather than by image or sequence complexity as it really prefers to do.

The *averaging period* used for defining the constant bit rate also has a major impact on video quality. For example, a stream with a CBR of "10 Mbps" could have a size of 10 Mb every second, or 5 Mb every half a second, or 100 Mb every 10 seconds. It is further important to note that the bit rate fluctuates within this averaging period. For example, we might be averaging 50 Mbps every 5 seconds, but this could mean 40 Mb in the first two seconds and 10 Mb in the remaining three seconds.

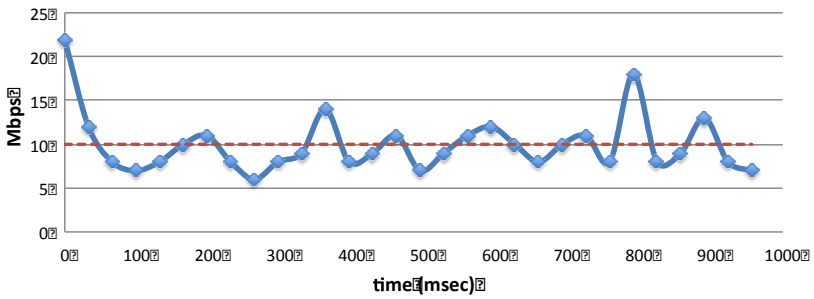

Figure 2: Example 10 Mbps CBR Stream with an Averaging Period of 10 Frames

Just as limiting the bit rate affects quality, limiting the averaging period also affects quality, with smaller averaging periods resulting in lower quality in the transmitted video.

Determining DSB and BRAB Buffer Sizes

Now we understand that a CBR stream actually fluctuates within the stream and that both the transmission bit rate and the averaging period affect quality. This allows us to determine how big the DSB and BRAB for a given system need to be.

First, appreciate that despite receiving data with a variable bit rate, the decoder will need to output data at a specific, really constant bit rate, as defined by the resolution and frame rate expected by the output display device (e.g., 1080p30).

If the communication channel between the encoder and the decoder has no bandwidth limitations and can transmit the fluctuating bit rates, then the decoder can begin |decoding as soon as it starts receiving the compressed data. In reality, the communication channel usually does have bandwidth limitations, for example 6 Mbps for 802.11b WiFi, or the video stream may be able to use only a specific amount of the available bandwidth as other traffic needs to go over the same channel. In these cases, the decoder would need to be fed data at rates that at times are higher or lower than the bit rate of the channel. Hence, the need for the DSB.

The DSB is responsible for bridging the communications rate mismatch and ensures that the decoder does not "starve" for incoming data, causing a playback interruption (recall the dreaded "Buffering …" message that sometimes appears when you're watching a NetFlix or YouTube video). The DSB achieves this by gathering and storing—buffering—enough incoming data until it can give the decoder enough data to process without any interruptions.

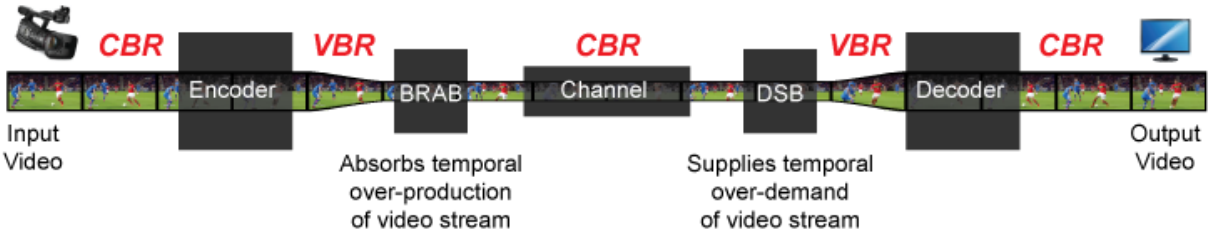

Figure 3: Video streaming over a bandwidth-limited channel, Constant and Variable Bit-Rates at different points.

The amount of buffering required depends on the bit rate and the averaging period of the stream. To make sure the decoder doesn't run out of data during playback, the DSB must store all the data corresponding to one complete averaging period. The averaging period—and therefore the latency related to the decoder's stream buffer—can range from a few tens of frames down to one whole frame, and in some cases, down to a fraction of a frame.

In a similar way, the BRAB's role is to normalize or average the data rate over the communication channel. Because the bit rate is actually fluctuating, there will be times that the compressed video bit rate will be higher than the bandwidth of our channel, and there will be times that it will be lower. To fully exploit our channel-bandwidth, the

BRAB needs to buffer an entire averaging period of the stream. Only in the presence of properly-sized BRAB and DSB buffers is uninterrupted live video streaming possible over bandwidth-limited communication channels.

In summary, because the DSB and BRAB have the biggest impact on end-to-end latency, and a CBR stream's averaging period determines the size of these buffers, it turns out that the averaging period is the most decisive factor in designing a low-latency system.

But how do we control the CBR averaging period?

Decreasing Latency with the Right Video Encoder

We've seen that while the size of the DSB greatly impacts latency, it's the rate control and averaging period definition occurring in the earlier video-encoding phase that actually determine how much buffering will be required. Unfortunately, choosing the best encoding for a particular system is not easy.

There are several encoding compression standards you may choose to use in a video system, including JPEG, JPEG2000, MPEG1/2/4, AVC/H.264, and HEVC/H.265. You would think these standards would include a specification for handling rate control, but none of them do. This makes the choice among standards a rather challenging task, and requires that you carefully consider the specific encoder in the decision-making process.

The ability to control the bit rate and the averaging period with minimum impact on video quality is the main factor that sets the best video encoders above the rest. A review of the available video encoding IP cores reveals quite a range in capability. On the less-than-great end of the spectrum are encoders with no rate-control capabilities, encoders that have rate control but don't offer enough user control over it, and encoders that support low-latency encoding, but at very different levels of quality.

Selecting the right encoder for a given application is a process involving video quality assessment and bit-rate analysis and is challenging even for expert video engineers. Non-experts such as typical SoC or embedded system designers should seek assistance from encoder vendors who should be able to facilitate and guide you through such an evaluation process.

Nevertheless, some key features can help you quickly separate efficient encoders from non-efficient ones, including *Rate-Control Granularity* and *Content-Adaptive Rate Control*.

Rate-Control Granularity

The rate-control process employs several sophisticated technical methods to modify the degree of compression to meet the target bit rate, such as quantization-level adjustment. Examining these methods is beyond the scope of this article, but a simple guideline can be applied—the more frequently the compression level is adjusted, the better the resulting compressed video will be in terms of both quality and rate-control accuracy.

This means that you can expect an encoder that does frame-based rate control such asregulating compression once every frame, to be less efficient than an encoder that makes rate-control adjustments multiple times during each frame.

So, when striving for low latency and quality, look for encoders with sub-frame rate control.

Content-Adaptive Rate Control

A single-pass rate-control algorithm decides on the right level of compression change based on knowledge and a guess. The knowledge is the amount of video data already transmitted. The guess is a predictive estimate of the amount of data needed to compress the remaining video content within the averaging period.

A smarter encoder can improve this estimate by trying to assess how difficult the remaining video content will be to compress, using statistics for the already compressed content and looking ahead at the content yet to be compressed. In general, these encoders with *content-adaptive algorithms* are more efficient, compared to content-unaware algorithms that only look at the previous data volumes.

Look for a content-adaptive encoder when both low latency and quality matter.

Practical Guidelines

We've seen that the need for data buffering increases video-system latency. We further understand that while some buffering occurs at the decoder (decompression) side, the factors actually influencing the amount of buffering necessary to meet transmission and quality goals are determined on the encoder (compression) side of the system.

When designing a system to meet low-latency goals, keep these points in mind:

- Achieving low latency will require some trade off of decreased video quality or a higher transmission bit rate (or both).
- Identify your latency contributors throughout the system, and eliminate any unnecessary buffering. Focus on the granularity level, such asframe, level, and pixel that matters most in your system.
- Make selecting the best encoder a top priority and evaluate each encoder's rate-control features. Make sure the encoder provides the level of control over latency that your system requires. At a minimum, make sure that the encoder can sup-port your target bit rate and the required averaging period.

Considering key encoder features like these can help you quickly create a selection short list. But, more so than with other IP cores, effective selection of a video encoder requires careful evaluation of the actual video quality produced in the context of the latency and bit rate requirements of your specific system. Be sure that you're working with an IP vendor who is willing to help you understand the latency implications within your specific system and who gives you a painless onsite evaluation process.

Reference Design: Ultra Low-Latency Video Streaming Over IP on Altera Arria V FPGAs

We have incorporated the guidelines for reducing video latency into a working reference design board. This system includes:

⇨ IP cores available from CAST:

- H.264 encoder core (sourced from technology partner Alma Technologies),
- H.264 decoder core (sourced from technology partner Fraunhofer HHI),
- CAST UDP/IP hardware stack
- CAST RTP hardware stack for H.264

⇨ Altera megafunctions:

- Hard DDR memory controller
- Triple-speed Ethernet MegaCore function (eMAC)
- HDMI MegaCore function

The system runs on two Arria V GX FPGA Starter Kits, one for the encoder and one for the decoder. The encoder board receives video via an HDMI receiver daughtercard from Terasic, Inc. The video is compressed by the H.264 encoder core, and the resulting NAL stream is optionally encapsulated in an MPEG-Transport or Real-time Transport Protocol (RTP) stream. The video data is subsequently packetized in UDP/IP packets and transmitted via Ethernet.

The FPGA on the decoder board receives, de-encapsulates, and decodes the incoming video stream. The decoded video drives a display via the HDMI MegaCore and the on-board HDMI transmitter. The entire design is controllable via UDPIP packets, allowing compression and other settings to be configured remotely from any network node.

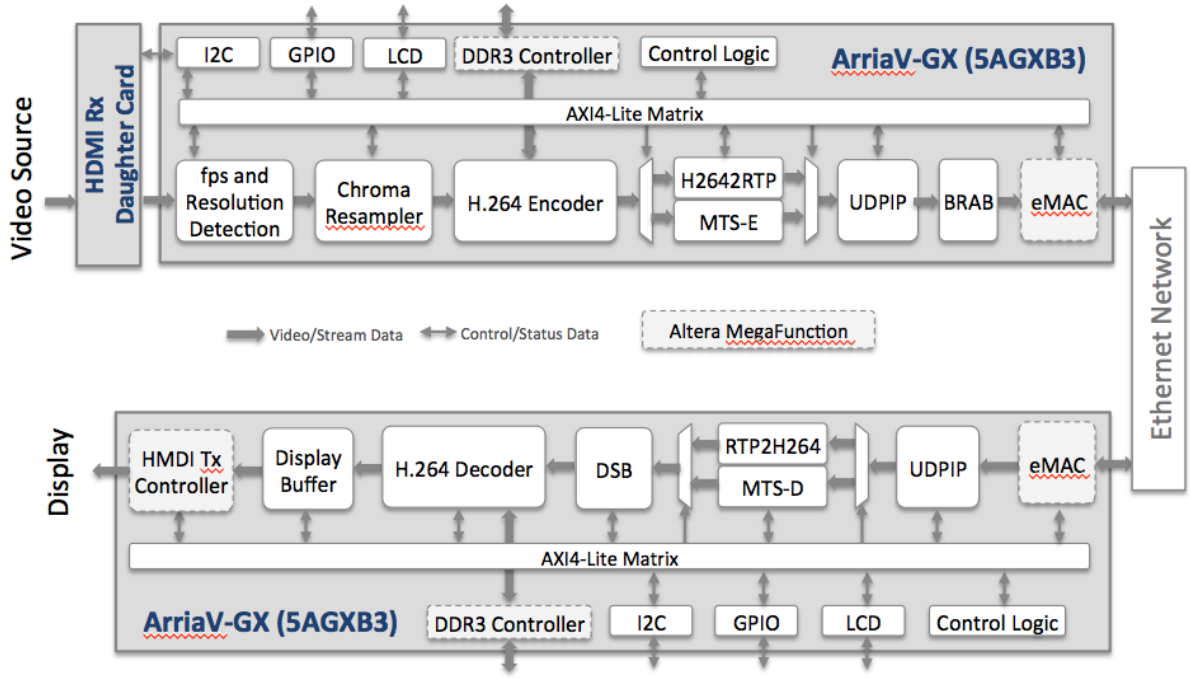

Figure 4: Arria V FPGA-Based Encoding and Decoding Systems.

The specific keys to achieving low-latency video in this design include:

⇨ The encoder implements Intra Refresh, allowing averaging periods down to 1/8 of a frame.

⇨ The decoder has a latency of just a few lines, unlike most decoders that would internally buffer one or more frames.

⇨ RTP/UDP/IP encapsulation is done by hardware stacks; using software stacks instead would impose further buffering and increase processing latency.

- ⇨ No display frame buffer is used.
- ⇨ The BRAB and DSB sizes are run-time programmable and set according to the H.264 encoder's rate control settings.

Table 2 shows the real-world results for 1080p@30 streaming video.

For this reference design, selecting the right video encoder and compatible IP enables extremely high-quality video streaming even at rates of 4 Mbps using the cost-effective Arria V FPGA.

Processing Stage	Buffering	Latency (1080p30)
Capture post-processing (chroma resampling)	Four lines	0.12 ms
Video compression	16 lines for conversion from raster scan and few thousand pixels on the encoder pipeline	0.5 ms
Bit-rate averaging buffer	¼ of a frame	8.33 ms
Network processing (RTP/UDP/IP encapsulation)	A few KB	<0.01 ms
Decoder stream buffer	¼ of a frame	8.33 ms
Video decompression	33 lines	1.02 ms
Display pre-processing (chroma resampling)	Four lines	0.12 ms
Display controller buffer	128k pixels	2.11 ms
Total	**Capture-to-display latency**	**20.54 ms**

Table 2. Latency Sources for the 1080p30 Video-Streaming System Ultra-Low Latency Reference Design

Consider the Video Compression Cores Available from CAST

Designing effective video processing and display systems requires considerable technical expertise, making IP selection challenging for most digital designers. At CAST, we strive to help you better understand low latency issues because we're confident that you'll then choose the IP solutions we offer, if they're the best fit for your needs.

We source video compression IP cores from partners who have been pioneering the fields of sophisticated, high-performance video and still-image compression, and provide them in high-quality, easy-to-use products ready for quick system integration. Complementary IP cores and design services available from CAST have been allowing our customers to develop successful video-streaming products using Altera devices for more than a decade. For more information, visit www.cast-inc.com/compression, or contact CAST at sales@cast-inc.com or +1 201.391.8300.