WHITE PAPER

Addressing the challenges of low latency video system requirements for embedded applications

abaco
SYSTEMS

Introduction

Video processing is ubiquitous in our everyday lives, and dates back nearly a century (electronic image transmission was first demonstrated 90 years ago). Transmission of images by digital means is 48 years old and commercial digital video 30 years old.

In recent years, improvements in video quality have skyrocketed; HD ("High Definition") video only within the 21st century and 4K video in the last 10 years. This has been a dramatic change. Consider that, in terms of pixels , HD represents a 200% - 575% increase, and full 4K video is a 4000% increase over the "standard definition" (SD) video employed up until 2000 (Super Bowl XXXIV was the first HD broadcast). As each pixel needs to be individually processed and transmitted, the need for the most powerful possible computing capability has risen in similarly exponential fashion, with a requirement (in the 60 frames per second case) to process over 18 Gigabytes of (unencoded) data per second – as opposed to 0.44 Gigabits/second for standard definition.

The challenges of handling this volume of data have been met with new processor technologies and systems designs. Graphics processing unit (GPU) technology, with its massively parallel capability, has been a huge enabler, delivering crisp, compacted HD and 4K+ video. However, a GPU can only process what it has in memory; getting that video into memory is another matter; the capture of this high-density video data in anything close to "real time" is a real problem - the "latency" problem.

Video systems are everywhere - from internet cat videos to medical imaging, military surveillance systems and rapidly-evolving autonomous vehicle applications. Digital video-based systems generally provide one or more of three, basic functions; image display (your TV set), image storage (for later analysis or viewing - like an MRI scan); or image data processing to extract information that directly initiates action (like the lane departure indicator on many modern cars).

In each case, the time it takes from a camera observing something in the real world (what we call "object space") to the time that observation is delivered electronically to the end use (display, storage or some action process) is called "latency". Latency has many components (such as transmission time to a remote location), but we will restrict this discussion to the latency experienced in the video processing chain.

In doing so, we will look at latency with regard to the video "use case". For example, a "live" TV show is not really "live" (that is; shown instantaneously) - indeed, the image you see is likely several seconds delayed (often deliberately long). Latency always exists simply due to the time needed to process the imagery through the electronic chain needed to deliver such beautiful pictures. No one thinks about or cares whether Odell Beckham Jr actually caught that ball one handed 500 milliseconds before they saw it on TV but that is definitely what happened.

Now: think about that same half second latency applied to alerting you that a car is rapidly approaching your car in your blind spot; that car has travelled three car-lengths or more in that time. If you are looking to change lanes, your blind spot detector needs to send that alert in 50 milliseconds, not 500.

Imagine how latency could affect a pilot using a video display in a degraded visual environment (fog, smoke, dust) or a surgeon using a video-guided surgical scalpel. This white paper will examine some applications where real-time video data latency is critical, and explore some of the system contributors to latency. It concludes with a review of some available COTS technology that addresses low latency video/ image processing requirements.

In systems where video data is presented to a human via a display, we note latency as "glass to glass" (that is to say the "glass" of the camera lens to the "glass" of the display). In other applications, latency is expressed as "glass to action" (the time it takes for the imagery seen by

The two most common video standards in the world are 60 frames per second (USA and many others) and 50 frames per second (UK, Europe and many others). Video images are composed of a series of "frames" (a frame being is composed of the entire image captured). In digital formats, standard definition video (SD), a frame is 640 pixels wide x 480 pixels high (a pixel being the smallest image element).

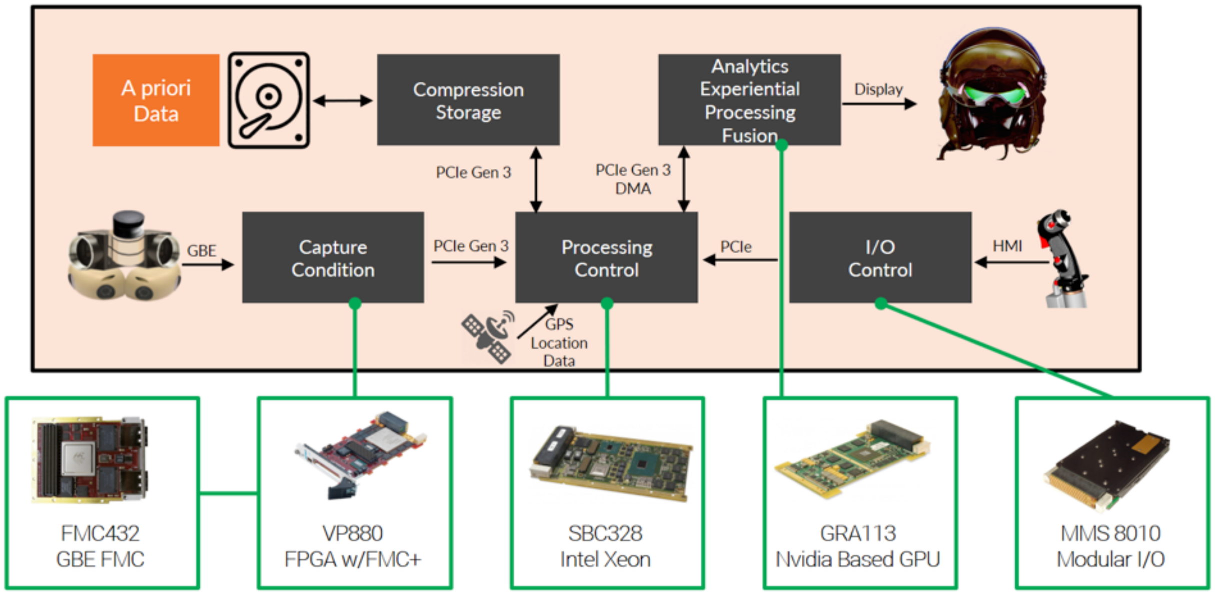

Figure 1: Simplified System Components for a sample Degraded Visual Environment System

the camera to be captured and processed to the point that it can initiate some activity, such as illuminate that little car indicator in your wing mirror).

Note: where video is stored, latency is generally not considered, except in the ability to store video at the rate it is collected. This paper will examine three image system types where the functions noted previously are used; degraded visual environment (DVE) visualization, autonomous vehicle operation, and active protection systems (APS) operation.

Video Latency Requirements for a Degraded Visual Environment Vision System

Tactical operations have always been at the mercy of the physical environment. Way back in time, it was cold, heat, terrain and so on that drove military requirements for equipment. In the 20th century, operations expanded to darkness, bad weather and so on. Today, operating sophisticated machines in extreme conditions, such as landing a helicopter in a "degraded" visual environment (darkness, fog, smoke, blowing sand, dust and snow) is essential for successful operations. These operations are extremely dangerous (indeed, according to one source, over 100 aircraft have been lost in the last 15 years due purely to loss of visibility and subsequent crashes).

To mitigate this danger, several DVEprograms have been established to equip these platforms with enhanced vision systems that utilize multiple sensors that "see" the environment in different spectral domains that penetrate obscurants; low-light TV (LLTV, operating beyond normal human vision) cameras; infra-red (IR) cameras; millimeter wave (MMW) radar and LIDAR. These sensors, situated outside the aircraft, provide video data/imagery that is processed into a composite (incorporating the best aspects of each sensor type) display for the operator. Because, in this case, the operator is not seeing "out the window", the resulting video display must be as "real time" as possible. Latency, then, is a critical design driver in DVE systems.

Figure 1 shows a simplified block diagram of the processing stages in such a system. On the "front end", video from a sensor (or sensors) is passed (over Gigabit Ethernet – GbE - in this case) to the "capture and condition" stage. (Most often, this is an FPGA processor). This pre-processor provides a very fast means of preparing and inputting video data to system memory via very fast Direct Memory Access (DMA) for use by a general purpose processor (GPP, sometimes called a CPU) and graphics processing unit where salient information is extracted from each sensor output and aggregated (often with associated, stored a priori data) into an operator-understandable display (called a "visualization").

Generally, images from the various image sources are "fused" into a single image with symbology overlays to indicate important features within the image. DVE systems have a typical latency requirement of 60ms glass-to-glass as a minimum - but less is very desirable. When considering the latency inherent in the sensor readouts, the display itself and the huge task of analyzing and fusing all this video, it is clear that this leaves little time for the actual capture (or "ingest") of the video at the "capture/condition" stage. Typically, this is 15ms or less.

Video Latency Requirements for Autonomous Vehicle Applications

It's not hard to see the similarities between the DVE system and autonomous vehicle sensor-based systems. Both require multiple sensors, operating in different spectral domains and (in the main) within semi- or non-cooperative environments that often are degraded by dust, fog and so on.

There is considerable debate about this figure. One famous study indicates that 80Ms is the highest value acceptable – based on studies using indirect vision hardware, others say that 60Ms (which happens to be the brain's "look-ahead" predictive processing threshold).

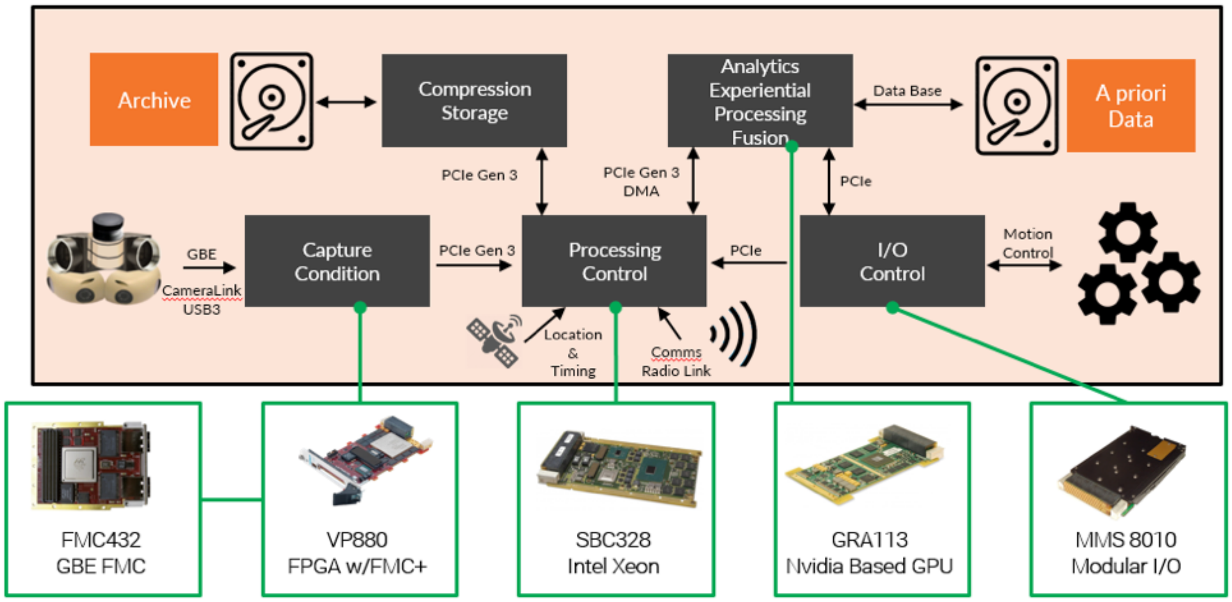

Figure 2: Simplified Autonomous Vehicle Video Processing Subsystem

The principle difference between the two is the fact that the DVE system provides a rich display to an operator that advises an action, whereas the autonomy system initiates that action itself. (There are also systems that provide semi-autonomous functions where only some vehicle functions are autonomous and others are dependent on the operator).

The consideration of latency is inextricably tied to the operational environment and mode. An aircraft (which operates in 3D space) has generally unobscured (albeit often degraded) lines of sight and usually is separated from objects by relatively large distances - so latency and operator reaction time form one equation. A vehicle travelling on a freeway at 80mph, a few feet away from another vehicle (which is operating according to an unknowable behavior) and where there are few clear lines of sight to potential hazards, represents another. More to the point, full autonomy requires the vehicle to take action itself. This theoretically eliminates the typical reaction time (500 ms, minus the brain's predictive 60ms) of a human that is largely the process of "deciding what to do". The autonomous system must work at least this well. Given that humans are really quite good at making decisions based on sparse data sets; have well "fused" sensors (sight, hearing, inertial) that are steerable; and have experience with predicting human behavior, the autonomous system must have more (and better sensors) to achieve the same situational understanding.

More sensors means faster ingest and processing. The "decision engine", utilizing a combination of sensor data (expressly not imagery – rather, data parsed from imagery) and learned, "experiential" programming (and likely some a priori data driven from stored maps) can make a decision faster than a human - but in order to make the right decision (in all circumstances) requires more data.

Some time is saved by this faster decision process, but some is lost in the additional sensor ingest and processing. On the plus side, again, the system does not have to waste time making the pretty picture that humans can understand; it simply analyzes the data and controls the vehicle accordingly. Given the use case, environment and operational scenario, a glass-to-action response time should look to be 40ms (about the time that vehicles with a closing speed of 80 mph can travel over 4 feet) to be effective.

This may seem a very small distance, but 40 ms is only the time needed to initiate the action, not the time to accomplish it. Vehicle response is a dominant factor; the vehicle must move and settle in a stable state for the maneuver to be successful.

Figure 2 shows a simplified block diagram of such a system, where a GbE sensor suite connects to an FMC-based interface to a 3U VPX FPGA processor - which in turn connects to a GPP and GPU working together to issue commands in an I/O module. This arrangement is architecturally very similar to that of a DVE application (note the elimination of the display and HMI (Human-Machine Interface) and addition of a control element).

Video Processing for Active Protection Systems

The third application example where real time low latency video is required is an active protection system (APS). Active protection systems provide protection to military vehicles by providing a counter to offensive weapons. Rather than surviving a "hit" by heavy armor, these systems either "spoof" or destroy incoming threats such as rocket-propelled grenades, mortars or gunfire before they hit. These systems utilize various sensors (essentially the same collection used in DVE and autonomy, but in different modalities) to detect, track and identify incoming projectiles and direct an appropriate counter response based on that knowledge.

The operation of these systems follows a very tight time line. For example, an RPG-7 (a common anti-tank weapon) has an effective range of about 1km. Assuming it is fused at 900m, its time-of-flight from launch to impact is about five seconds.

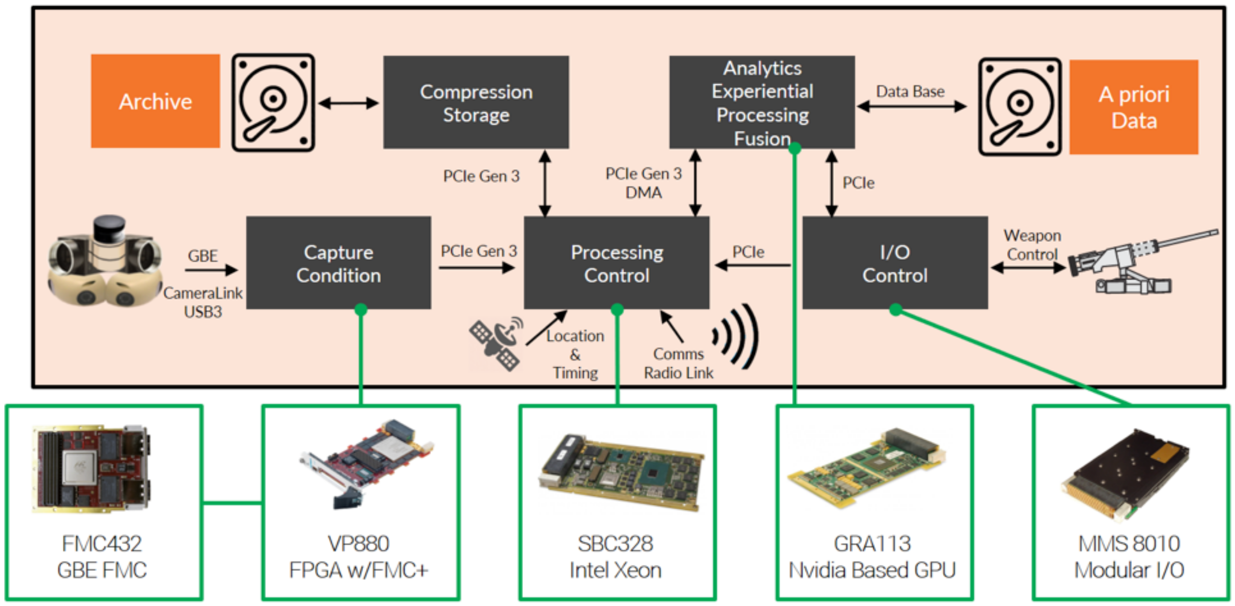

Figure 3: Simplified Active Protection Video Response system

Presuming one second for detection, correlation and confirmation by multiple sensors and establishment of a track in another 100ms or so, the projectile will have traveled about 300 meters (one third the distance to the target). The initiation of a counter weapon and its launch must then occur very quickly (and in cases where active tracking of both the incoming and outgoing ordinance takes place, a very fast control loop is required).

If the protective response takes one second (including time of flight for the counter weapon), the incoming projectile would be dangerously close indeed (easily within 300 meters) and that's the easy case; imagine a threat that is a half or a third of that distance - or a much faster weapon.

Latency in this system is obviously demanding and glass-to-action (counter weapon initiation) is typically under 25ms (the time it takes the RPG to travel about 300 feet). An active-protection processing architecture is like that shown in earlier block diagrams; however, here the system is totally non-imaging but must do the additional work of threat identification, counter weapon selection, precision tracking and kill assessment.

In all three of these systems, GPU or GPP processing merging real-time data with a priori data is required to speed visualization generation and/or the decision processing. In each case, experiential (cognitive) processing is to be expected to be present.

Contributors to Video Latency in System Design

As we have seen, the basic architecture of many video/image processing subsystems is remarkably similar; sensor data/imagery is ingested and pre-processed in a "capture/condition" stage. Salient data is extracted in this stage and sent via high-speed interface to the analytical stage where suitable visualizations or action initiations are created. The computations undertaken in these stages (and in which stage they are conducted) vary with the application, but often include; compression, feature extraction, de-warping, contrast enhancement, optical flow, edge detection, cross domain correlation, motion tracking and image and data fusion to name a few.

In a properly designed subsystem, these operations are hosted in the stage best suited for the purpose. For example; any computation that reduces the bit rate of the data flow (such as Bayer encoding) should be as close to the source as possible. A keen focus on the specific data that is most important to the functionality needed is essential to avoid processing "un-interesting" data and wasting precious processing resources, leading to increased latency. Clearly, latency is affected by the application of computing resources to algorithms - but it is also impacted by the type of processor and transfer technology employed.

Compression Latency

Due to bit rate and volume, many video systems utilize compression techniques to reduce data bandwidth or storage size. The most common of these is video (as in MPEG) compression - algorithms that take advantage of image features such as a static background that can be represented with fewer bits.

Usage of compression algorithms makes the data size dependent on the image content and results in a variable bit rate (VBR) for transfer. Variable bit rate systems require a data transport mechanism to be designed for the worst-case bandwidth requirements, otherwise a system may drop data or exhibit buffering and variable latency. In some systems, a maximum data transport bandwidth is established and compression algorithms must sacrifice video quality to fit within these constraints.

Video compression algorithms may require a minimum data set such as a full line or frame to start processing; thus, a certain amount of buffering may be required, adding to total system latency. In our three examples, it may be impractical to employ certain compression algorithms (especially ones in software as these are known to add significant latency). However, some compression could perhaps be implemented with a hardware streaming architecture inside the pre-processor FPGA such that time lost in compression is gained in reduced transfer time.

Another form of compression is data compression. In many applications and functionalities, it is not necessary to process the entire image produced by a camera. Just as MPEG compression only transfers content that changes frame-to-frame, data compression only passes "interesting" content from the frame. This could be motion (vector, velocity), detection of a contiguous group ("blob") of pixels, the edges of objects in the frame or the point-cloud generated by a LIDAR.

Thus, it can be readily seen that in systems that do not require a display (and the resulting visualization generation) data compression is dominant and video compression (except for non-real-time archival recording) is not. In cases where visualization is needed, video compression may be dominant. However, in the cases mentioned, data compression is always performed in one form or another.

Transfer Latency

As noted, the three applications discussed have three different use cases which require a variety of processing functionalities; streaming, pre-processing, threat/obstacle detection, video fusion, and so on. Where in the architecture these functionalities are performed is important in optimizing latency performance. (Indeed, the expression 'use the right tool for the job' applies here). The most appropriate processing tool for each task; FPGA, GPP, and GPU is employed where it is most effective.

While there is a trend to converge these three processing elements, today, it is generally most economical to consider these as two or three separate cards in a system. In our representative subsystem designs, we have chosen PCIe Gen3 as our element-to-element data fabric for its high data rate (slower fabrics are intolerably latent). Latency is also reduced by leveraging PCIe Gen3 DMA (wherein data is written directly to memory).

Processing Latency

The central piece of the processing control system is the general purpose processor. The GPP can provide a variety of functions; data may be correlated here, analyzed to discriminate obstacles or threats or the GPP may simply be a host controller for faster pre-process and GPU stages. This sort of decision based processing can require significant multi-threaded parallelism, analyzing multiple scenarios and data sets.

Where real-time deterministic operation is needed, selection of the appropriate GPP is essential. The Intel® Xeon® processor featured on Abaco's SBC328 (shown) 3U VPX single board computer has significant dedicated parallel cores, providing ample power and parallelism for these demanding applications.

The GPU stage is generally where the "heavy lifting" takes place. It is here where "learning" and image fusion takes place, where Visualizations are generated and where data is ultimately correlated, analyzed and reduced to action. Generally, 'the more cores the better' in this stage and here we show Abaco's GRA113 GPU with 640 cores supporting v3.0 CUDA™ (as well as OpenGL®) and a 16-lane PCIe™ interface.

Code Latency

Programming is obviously the most crucial part of the design. The optimum hardware implementation cannot fully make up for non-optimized code. Of course, it is the use case that dominates the software employed but, as noted, many functionalities, such as edge detection, de-warping and fusion are common. While the algorithms behind functionalities are readily available (including on-line) writing efficient code to implement them can be tedious, time-consuming and fraught with inefficiencies.

Not only are these inefficiencies creators of latency, they can also prevent code from being certifiable to safety standards (clearly important in all the cases described herein). Abaco has developed a series of tools to assist in the development of code for complex computer architectures called AXIS. For image processing, visualization and graphics, we offer AXIS ImageFlex™, a library of optimized code suitable for direct incorporation into higher-level applications. AXIS ImageFlex reduces the time and inefficiency associated with many functionalities often by 40-50% in execution speed and lines of code by a huge 500%.

Other tools in the AXIS family work to analyze and optimize multi-processor architectures.

Benefits of Building Video Systems on an Adaptive FPGA-Based Modular COTS Approach

Leveraging commercial technology can have significant benefits in computer systems; development costs are reduced, upgradability is improved, and obsolescence management is simpler.

The three applications presented here are highly demanding - not only in terms of processing latency, but also because these applications are evolving constantly, accommodating new sensors and requiring new functionalities. Leveraging an extremely adaptable platform like 3U VPX allows performance equal to bespoke designs in a COTS approach.

Nowhere in these systems is this more evident than in the pre-processing stage, with the inclusion of a powerful FPGA employing a modular sensor interface. By leveraging FPGA technology, these systems are readily adaptable to diverse I/O interfaces and protocols utilizing technologies like VITA 57.4 defined FPGA mezzanine cards (FMC) and Abaco's patented Micro Mezzanine System (MMS). On the sensor processing side, products like the powerful VP880 and the FMC432 enable 10 Gigabit Ethernet to connect to Xilinx®'s most powerful Ultrascale™ processing chip via the standard FMC+ interface.

This 3U VPX board, built on the power and flexibility of the OpenVPX standard, is capable of PCIe Gen3 and has enabled dedicated peer-to-peer DMA links from card to card meaning low, predictable latency transfer of large data sets. In terms of the general purpose processors and graphics processing units, Abaco provides the most up-to-date processor technology based on these form factors.

Abaco Systems, within the framework of open standards, provides C OTS hardware, design tools and application knowledge to fully support development, deployment and lifetime support of these critical applications.

Conclusion

As we have seen management of latency in systems reliant on video sensors is a critical design driver. It is only through the use of the right tools and techniques that latency challenges can be overcome. An optimum design can be achieved through the use of powerful COTS processors, connected via efficient interfaces and running efficient software. Abaco Systems provides these elements and has the expertise to assemble these elements within higher-level systems to achieve end performance goals.

AXIS

ImageFlex Is a trademark of Abaco Systems. CUDA is a trademark, of NVIDIA Corporation. Intel and Xeon are registered trademarks of Intel Corporation. Xilinx is a registered trademark, and Ultrascale is a trademark, of Xilinx Inc. OpenGL is a registered trademark of Silicon Graphics Inc. PCIe is a trademark of PCI-SIG. All other trademarks are the property of their respective owners.