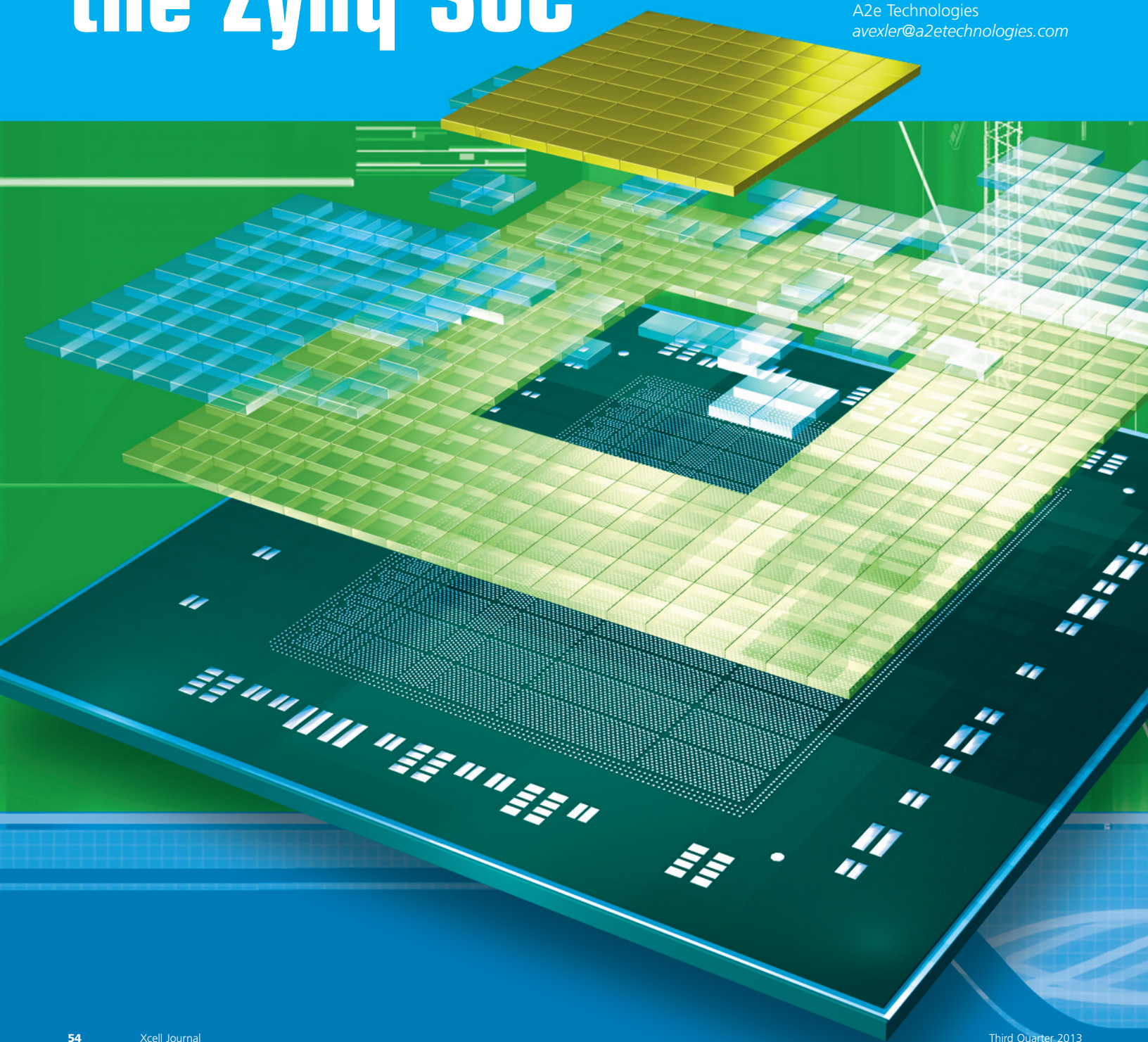
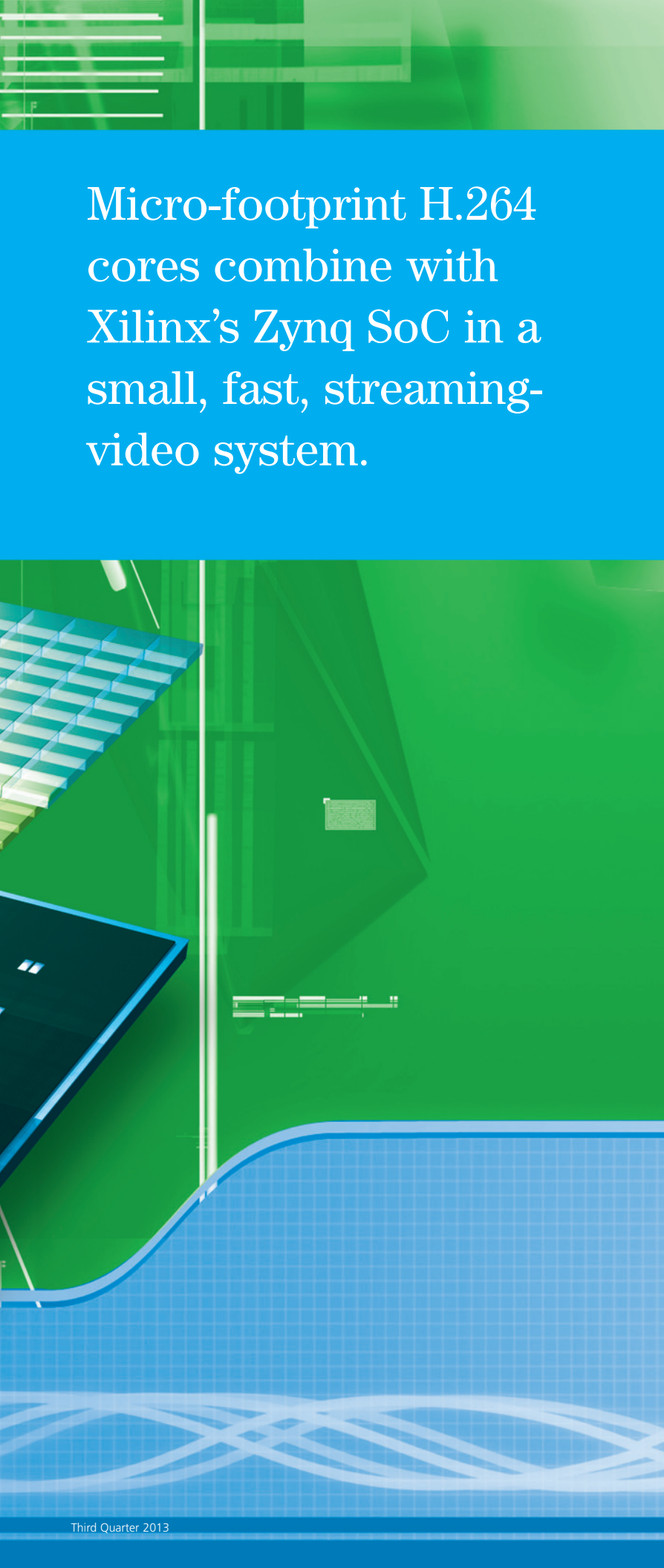


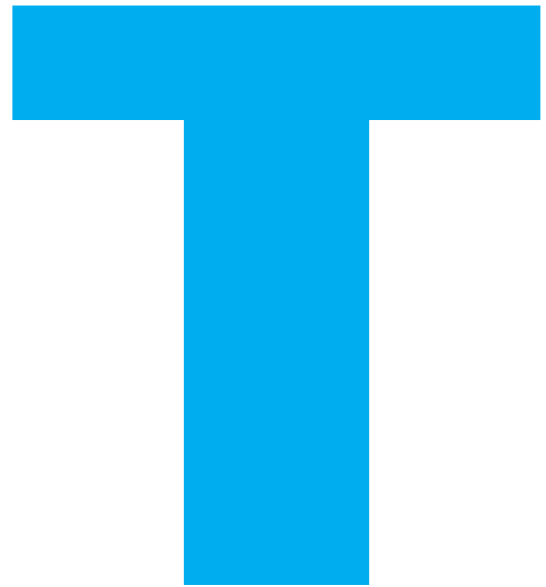
Designing a Low-Latency H.264 System Using the Zynq SoC

by Allen Vexler
CTO
A2e Technologies
avexler@a2etechnologies.com





Micro-footprint H.264 cores combine with Xilinx's Zynq SoC in a small, fast, streaming-video system.



To create an IP-based streaming-video system with a small PCB footprint, designers historically have had to choose between the inflexible architecture of an ASSP or a larger, more flexible system based on an FPGA-microprocessor combo. Placing a soft-core microprocessor inside the FPGA could eliminate the need for a separate processor and DRAM, but the performance of the final system might not match that of a solution built around an external ARM® processor that would also contain USB, Ethernet and other useful peripherals. With the advent of the Xilinx® Zynq®-7000 All Programmable SoC along with small-footprint H.264 cores, it's now possible to construct a system in a very small-form-factor PCB with only one bank of DRAM, but with the power of dual ARM cores and high-speed peripherals that are all interconnected with multiple high-speed AXI4 buses (see Figure 1).

Although H.264 cores for FPGAs have existed for quite some time, up until now none have been fast enough and small enough to convert 1080p30 video and still fit into a small, low-cost device. Using the latest micro-footprint H.264 cores from A2e Technologies in tandem with the Zynq SoC, it's possible to build a low-latency system that can encode and decode multiple streams of video from 720p to 4K at variable frame rates from 15 to 60 frames per second (fps). Placing an A2e Technologies H.264 core in a Zynq SoC device allows for a significant reduction in board space and component count, while the dual ARM cores in the Zynq SoC remove the need for a separate microprocessor and the memory bank to which it must be attached.

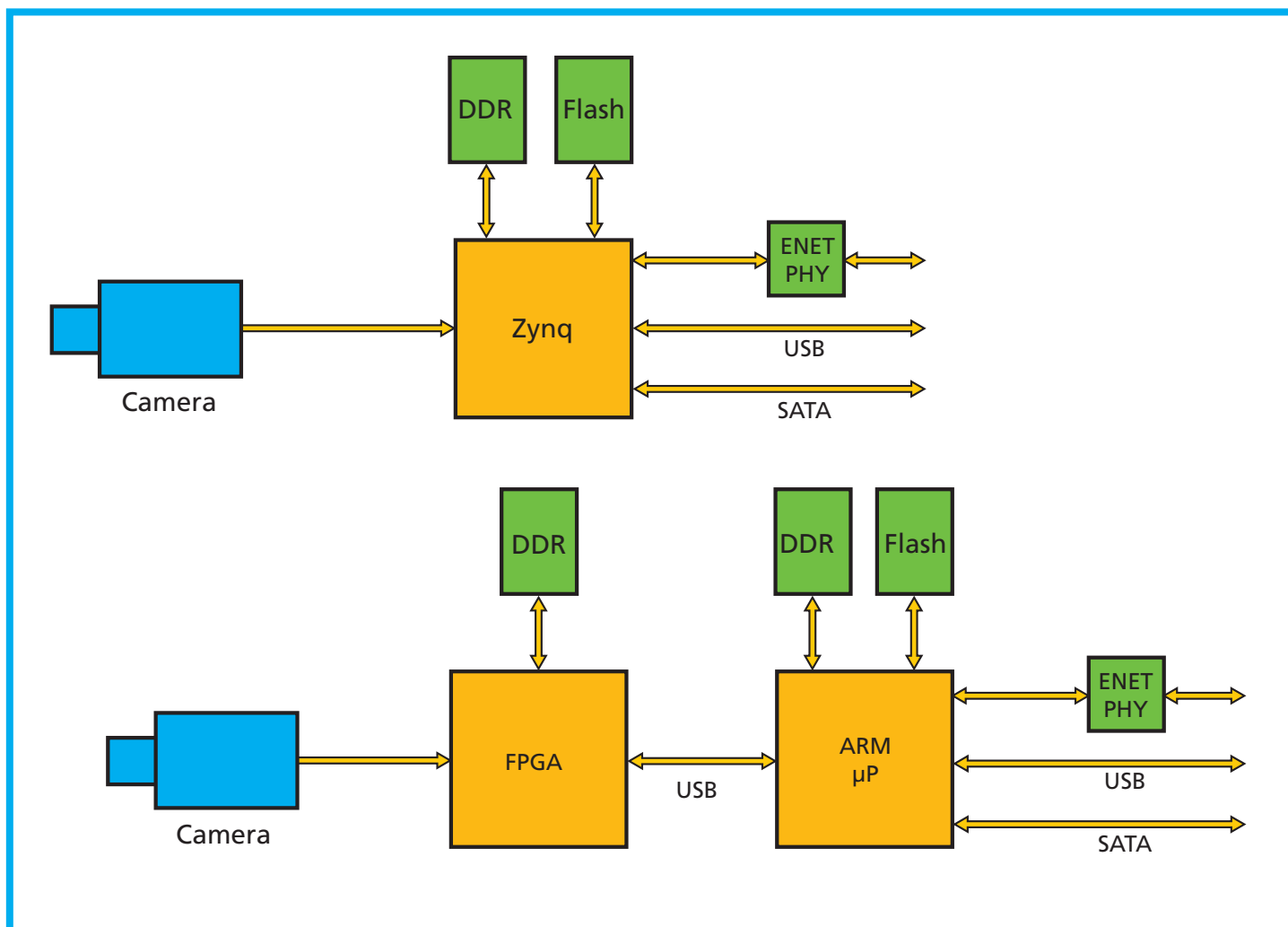


Figure 1 – Video compression systems based on a single-chip Zynq SoC (top) vs. a two-chip processor-FPGA combination

The result is a complete streaming-video system, built around Linux drivers and a Real Time Streaming Protocol (RTSP) server, that is capable of compressing 1080p30 video from two cameras with an end-to-end latency of approximately 10 milliseconds. The A2e Technologies H.264 core is available in encode-only and encode/decode versions. Additionally, there is a low-latency version that reduces encode latency to less than 5 ms. The 1080p30 encode-only version requires 10,000 lookup tables (LUTs), or roughly 25 percent of a Zynq Z7020's FPGA fabric, while the encode/decode version requires 11K LUTs.

SOC-FPGA BASED SYSTEMS

Products based on the Zynq SoC provide significantly more flexibility than those built using an application-specific standard product (ASSP). For example, it's easy to construct a system supporting four 1080p30 inputs by placing four H.264 cores in the FPGA fabric and hooking each camera input to the FPGA. Many ASSPs have only two inputs, forcing designers to figure out how to multiplex several streams into one input. Each A2e Technologies H.264 core is capable of handling six VGA-resolution cameras or one camera with 1080p30 resolution. So, it would be possible to construct a two-core sys-

tem that could compress video from 12 VGA cameras.

ASSPs typically provide for an on-screen display feature only on the decoded-video side, forcing designers either to send the OSD information as metadata or to use the ARM core to time video frames and programmatically write the OSD data to the video buffer. In an FPGA, adding OSD before video compression is as simple as dropping in a block of IP. You can place additional processing blocks such as fisheye-lens correction before the compression engine with relative ease. FPGAs also allow for in-field, future upgrades of features, such as adding H.265 compres-

sion. Figure 2 is a block diagram of an H.264 compression engine with input from two 1080p30 cameras, with OSD applied to the uncompressed image.

MANAGING LATENCY

Certain applications, such as control of remotely piloted vehicles (RPVs), are based upon feedback from streaming images coming from the remote device. In order to control a remote device, the latency between the time when the sensor sends the video to the compression engine and the point when the decoded image is displayed (referred to as “glass to glass”) typically needs to be less than 100 ms. Some designers target num-

bers in the 50-ms range. Latency is the sum the following:

- Video-processing time (ISP, fisheye-lens correction, etc.)
- Delay to fill frame buffer
- Compression time
- Software delay associated with sending out a packet
- Network delay
- Software delay associated with receiving a packet
- Video decode time

Many systems encode video using hardware but end up decoding it using a standard video player such

as VLC running on a PC. Even though the buffer delay for media players can be reduced from the typical 500 to 1,000 ms to something significantly less, the latency is still far greater than 50 ms. To truly control latency and keep it to an absolute minimum, hardware decode of the compressed video stream is required, along with minimal buffering.

Latency for H.264 encoders is typically specified in frames, as a full frame must typically be buffered before compression begins. Assuming you can speed up your encoder, you could decrease the encode latency simply by doubling the frame rate—that is, one frame of latency for 30 fps is 33 ms vs. one frame of latency for 60 fps is 16.5

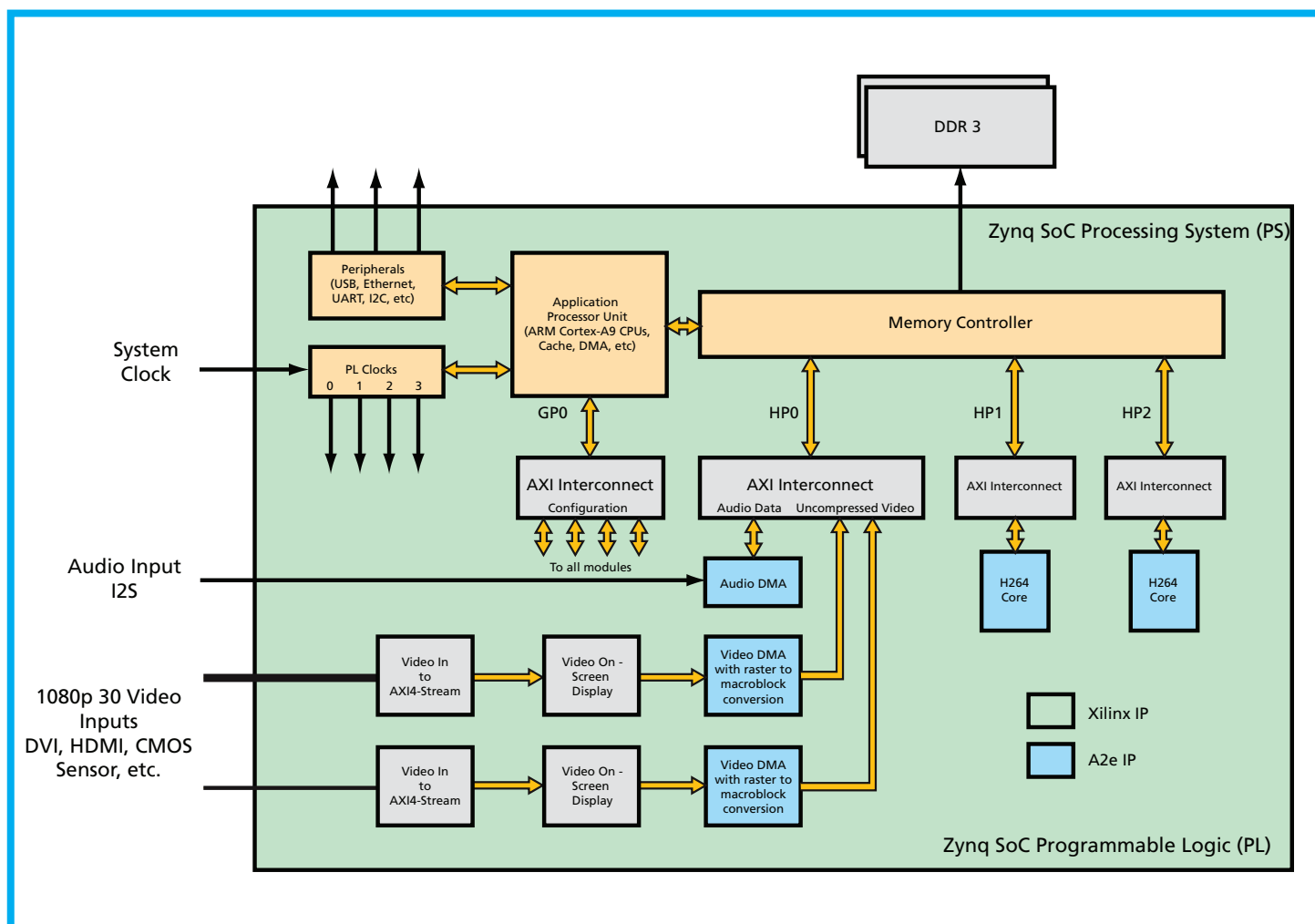


Figure 2 – Architecture of a dual-1080p30 H.264 compression engine

A typical streaming-video system uses an RTSP server to establish a streaming-video connection between a camera and the client (decoding/recording) device. The RTSP server will stream the compressed video to the client for display or storage.

ms. Many times this option is not available due to performance limitations of the camera and encoder. Thus, the solution is an encoder specifically designed for low latency. In the new A2e Technologies low-latency encoder, only 16 video lines need to be buffered up before compression starts. For a 1080p30 video stream, the latency is less than 500 μ s. For a 480p30 video stream the latency is less than 1 ms. This low-latency encoder will allow designers to construct systems with low and predictable latency.

In order to minimize total latency, you must also minimize delays introduced by buffers, network stacks and

RTSP servers/clients on both the encoding and decoding sides, as it is pointless to use a low-latency encoder with a software path that introduces a large amount of latency. An RTSP server typically is used to establish a streaming-video connection between a server (camera) and the client (decoding/recording) device. After a connection is established, the RTSP server will stream the compressed video to the client for display or storage.

MINIMIZING LATENCY

Typically, the software components in the server and client are designed only to keep up with the bandwidth required to stream compressed video,

nor minimize latency. With a non-real-time operating system like Linux, it can be difficult to guarantee latency. The typical solution is to create a low-latency, custom protocol for both the server and client. However, this approach has the downside of not being compatible with industry standards. Another approach is to take a standard like RTSP and make modifications at the lower levels of the software to minimize latency, while at the same time retaining compliance with standards.

However, you can also take steps to minimize copies between kernel and user space, and the delays associated with them. To reduce latency though

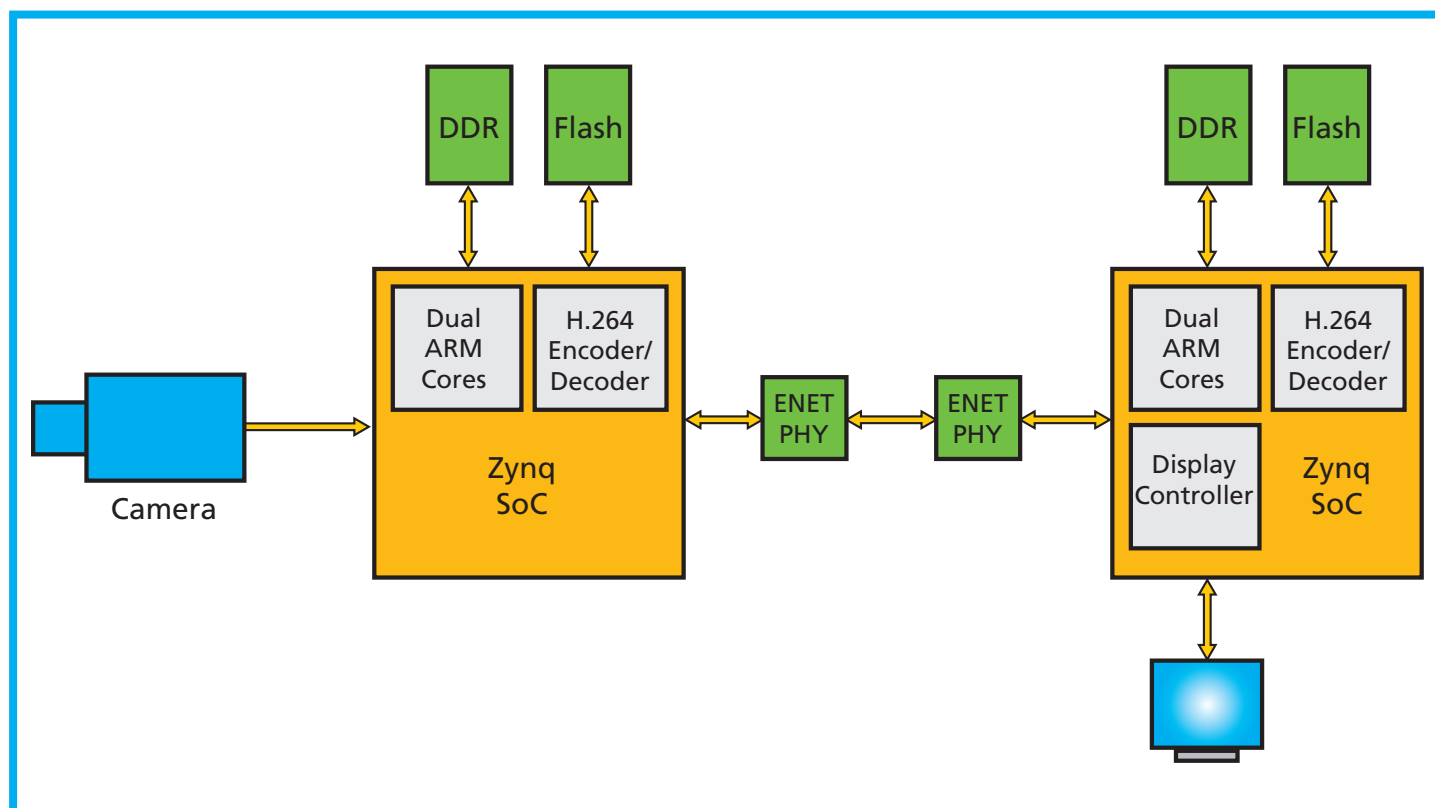


Figure 3 – Block diagram of complete encode-decode system

the whole software path, you will need to decouple the RTSP server and compressed data forwarding so that the Linux driver, not the RTSP server, performs the forwarding.

In the A2e Technologies low-latency RTSP server, we have made two major changes to reduce latency. First, we removed the RTSP server from the forwarding path. The RTSP server continues to maintain statistics using the Real Time Control Protocol (RTCP), and periodically (or asynchronously) updates the kernel drive with the changes in the network destination (i.e., destination IP or MAC address). Second, the kernel driver attaches the necessary headers (based on the information pro-

vided by the RTSP server) and immediately forwards the packet by directly injecting it into the network driver (for example, `udp_send`), thus eliminating the memory copy to/from user space.

Figure 3 shows a complete H.264 IP-based encode/decode system with a combined latency of less than 50 ms. The system is based upon the Zynq SoC, the A2e Technologies low-latency H.264 encoder-decoder and the A2e Technologies low-latency RTSP server/client. Note that the only real difference from a hardware perspective between the encode and decode systems is that the encode side must interface to a camera/sensor and the decode side

must be able to drive a flat-panel display. You can easily design a board that has all required hardware features needed for use in both encode and decode.

In order to minimize video compression/decompression latency for real-time control applications, designers need special encoders and optimized software. Utilizing the Xilinx Zynq SoC and the A2e Technologies low-latency H.264 encoder and optimized RTSP server/client, you can create a highly configurable system with extremely low latency in a small PCB footprint.

For more information, visit <http://www.xilinx.com/products/intellectual-property/1-1LK9J1.htm> and <http://www.a2etechnologies.com>.



Stop soldering down expensive FPGAs and ASICs without sacrificing performance. New GHz Universal BGA sockets from Ardent provide the ultimate convenience and cost savings for your development project.

Up to 2500 I/Os. Under \$600 each, hardware included.

www.ardentconcepts.com 603.474.1760



US Patent Numbers 6,787,709, 6,909,056, 7,126,062, 7,556,503. Other US & Foreign Patents Pending. Copyright 2012 Ardent Concepts.