

Encoding High-Resolution Ogg/Theora Video with Reconfigurable FPGAs

Once the traditional application area of custom ASICs, modern FPGAs can now handle high-performance video encoding.

by Andrey Filippov
President
Elphel, Inc.
andrey@elphel.com

Much of the Spring 2003 issue of the *Xcell Journal* in which my article about Spartan™-IIE-based Elphel Model 313 cameras appeared (“How to Use Free Software in FPGA Embedded Designs”) was dedicated to the Xilinx® Spartan-3 FPGA. I immediately started to think about using these devices in our new generation of Elphel network cameras, but it wasn't until last year that I was finally able to start working with them.

One of the factors that slowed my company's adoption of this new technology was the fact that at first I could not find appropriate software that could handle the devices selected, as it is essential that our end users can modify our products without expensive software development tools. When I visited the Xilinx website in Summer 2004 and found that the current version of the free downloadable WebPACK™ software could handle the XC3S1000 – the largest device available in a small FT256 package – I knew it was the right time to switch to the Spartan-3 device.

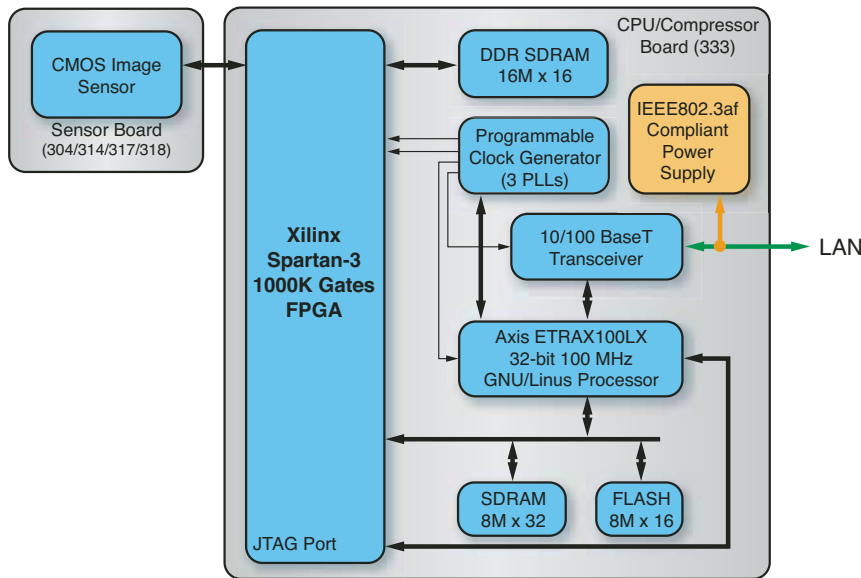


Figure 1 – Camera system block diagram

The Camera Hardware

The new Model 333 camera (Figure 1) uses the same Linux-optimized CPU (ETRAX100LX by Axis Communications) as the earlier Model 313, but with increased system memory – 32 MB of SDRAM and 16 MB of Flash. The second major upgrade is the use of 32 MB of DDR SDRAM as a dedicated frame buffer that works in tandem with the FPGA, supplementing its processing power with high capacity and I/O bandwidth.

The Spartan-3 DDR I/O functionality made it possible to increase the memory bandwidth without increasing board size – the complete system still fits on a 1.5 x 3.5-inch four-layer board (see Figure 2). The actual board area is even smaller, as the new one is designed to fit the sealed RJ45 connectors for outdoor applications.

For the camera circuit design, the goals include combining high computational performance with small size (that also simplifies preserving high-speed signal integrity on the PCB) and providing the flexibility for the reconfigurable FPGA on the system level. For the latter, I decided to split the camera circuitry into two boards: one main board and a second containing just a sensor with minimal related components. On the main board the FPGA I/O pins go directly to the inter-board connector, so it is possi-

ble to change the pin functions (including polarity) to match the particular sensor boards. A similar solution allowed the earlier Model 313 camera to support different types of sensors (most became available after the board design). It even works in our 11-megapixel Model 323 cameras without any PCB modifications.

Selecting the Video Encoding Technique

After the prototype camera was ready, it took just a couple of weeks to modify the code developed for the Spartan-IIE-based



Figure 2 – Camera system board

camera and to implement motion JPEG compression. Half of that time was spent trying to figure out how to configure the new FPGA with the generated bitstream. In the camera, JTAG pins of the device are

connected directly to the processor I/O pins, so I could not use the software that comes with Xilinx configuration hardware. The JTAG instruction register is six bits wide, not five as it was in the Spartan-IIE devices with which I was familiar. After some trial and error, I figured that out and found that the same code could run at 125 MHz (instead of 90 MHz in the previous model) and used just 36% (not 98% as before) of available slices – plenty of room for more challenging tasks.

Of course, I had some challenging tasks in mind, as motion JPEG is not a really good option for high-resolution/high-frame-rate cameras because the amount of data to be transferred or stored is quite huge. It is a waste of network bandwidth or hard disk space when recording such video streams, as fixed-view cameras in most cases have very little difference between consecutive frames. Something like MPEG-2 could make a difference; that was the standard I was planning to implement in the camera.

But as soon as I got some books on MPEG-2 and started combing through online resources, I found another fundamental difference between MPEG and JPEG – not just that it can use the similarity between consecutive frames. Contrary to JPEG, MPEG-2 requires you to pay licensing fees for using the encoders based on this standard. The fee is small compared

to the cost of the hardware, but it still could be a hassle and does not provide freedom for implementation.

It did not take long to find a perfect alternative – Theora, based on the VP3

codec developed by On2 Technologies (www.on2.com) and released as open-source software for royalty-free use and modifications (see www.theora.org/svn.html).

Theora is an advanced video codec that competes with MPEG-4 and other similar low-bit-rate video compression schemes. It is now supported by the Xiph.org Foundation along with Ogg, the transport layer used with Theora to deliver the video content. The bitstream format is stable enough and supported by multiple players running on different operating systems. Like JPEG and MPEG, it uses a two-dimensional 8 x 8 DCT.

FPGA Implementation

The code for the Elphel Model 333 camera FPGA is written in Verilog HDL (Figure 3). It is designed around the 8-channel SDRAM controller that uses the Spartan-3 DDR capabilities. The structure of the memory accesses and specially organized

data mapping both serve the same goal: optimizing memory bandwidth that otherwise would be a system bottleneck.

The rest of the code that currently uses two-thirds of the general FPGA resources (slices) and 20 of 24 block RAM modules includes video compression modules, a sensor, and system interfaces.

A detailed description of the camera code is available, together with the source code, at Sourceforge (<https://sourceforge.net/projects/elphel>).

Conclusion

High-performance reconfigurable FPGAs made it possible to build a fast high-resolution low-bit-rate network camera capable of running 30 fps at a resolution of 1280 x 1024 pixels (12 fps at a resolution of 2048 x 1536). Many of the new features of the Spartan-3 devices proved to be very useful in this design: embedded multipliers for DSP functions, advanced digital clock

management, DDR I/O functions, an increased number of global clock networks for the DDR SDRAM controller, and large block RAM modules for the various tables and buffers in the camera.

The free video encoder (Theora) and completely open implementation of the camera (all software and Verilog code is provided under the GNU General Public License) makes the second most important function of Elphel products possible. You can use these cameras not only as finished products but also as universal development platforms – demonstrating the power and flexibility of the Spartan-3 family. It is possible to add your own code, rerun the tools (both for the FPGA code and the C-language camera software), and immediately try the new camera with advanced image processing implemented.

For more information, visit www.elphel.com, <https://sourceforge.net/projects/elphel>, and www.theora.org.

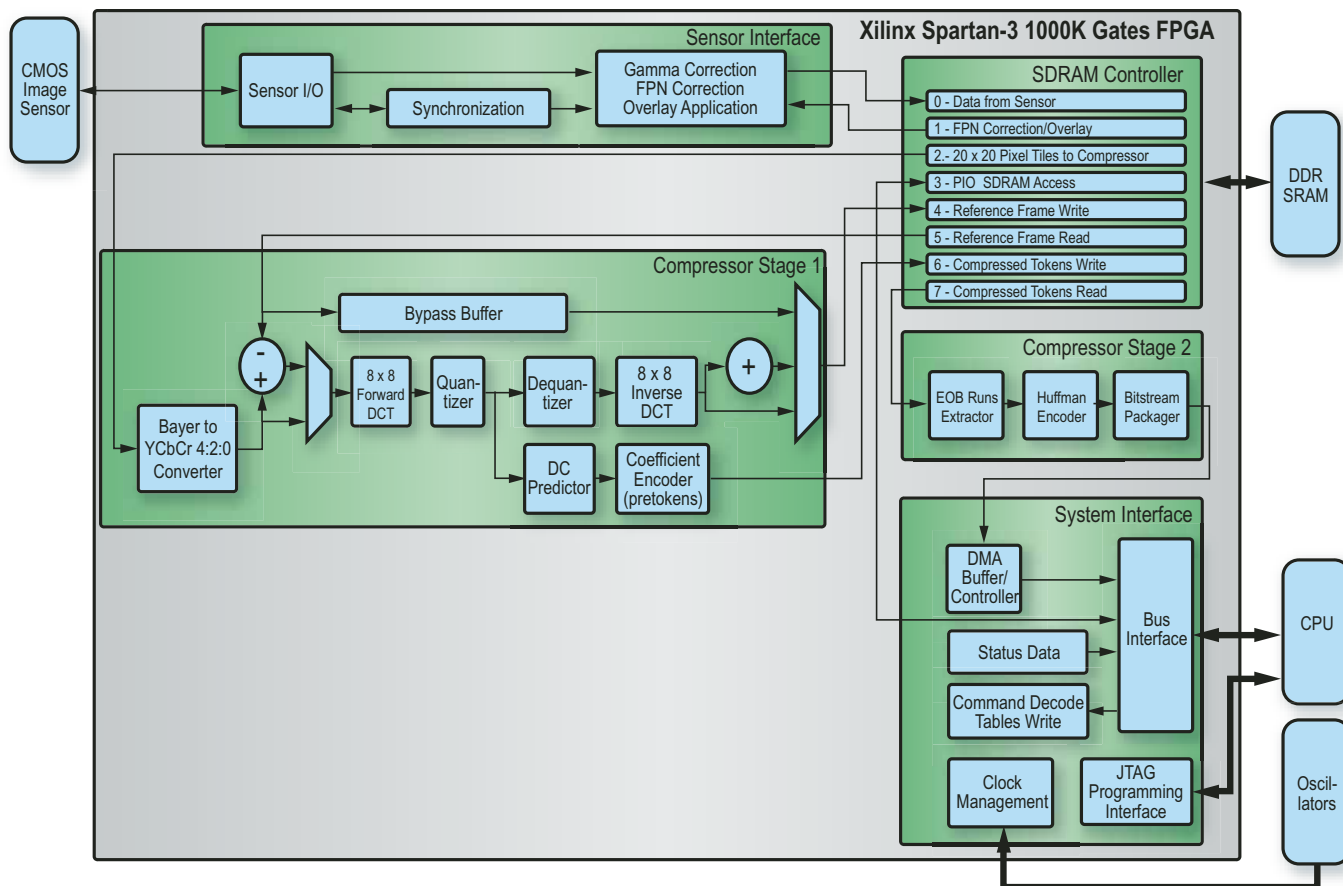


Figure 3 – Block diagram of the FPGA code