SoC drawer: SoC design for hardware acceleration, Part 2

Computation-hungry applications for offload

Sam Siewert (Sam.Siewert@Colorado.edu), Adjunct Professor, University of Colorado

Summary: In the SoC design for hardware acceleration series, author Sam Siewert migrates a simple C function to a SystemC specification that can be simulated and verified for ultimate implementation as a hardware function. Part 1 provided the C code and a general overview of video capture, streaming, and processing. Part 2 shows how hardware acceleration of emergent applications, such as video streaming, can benefit from system-on-chip (SoC) design and reconfigurable SoCs with hybrid C software and field-programmable gate array (FPGA)-based functionality.

View more content in this series

Date: 22 Aug 2006
Level: Introductory
Activity: 918 views
Comments: 0 (Add comments)

Average rating (based on 3 votes)

As Part 1 of this series discussed, SoC designs offer the opportunity to migrate functionality initially implemented in software and firmware into hardware acceleration state machines. Reconfigurable SoC platforms like the Xilinx Virtex-4 provide the capability to implement functions on PowerPC® chips as C code and to accelerate key functions with offload to the FPGA fabric.

Video streaming is an exciting application for a hybrid reconfigurable SoC that uses software and hardware state machines. Many emergent applications, such as Web-based geographic information systems (GISes) like Google Earth and Microsoft® Virtual Earth, promise to transform the way we see the world through the Web, but they require significant real-time image processing, storage, advanced search, and a host of potentially hardware-accelerated features on the server and client sides to really take off.

This article examines a working example of how reconfigurable SoCs can help accelerate exciting new emergent applications and Web 2.0 features, taking the basic C function for enhancing an image developed in Part 1 and offloading it to a SystemC specification.

Use of a hybrid reconfigurable SoC for hardware acceleration of media processing

The Xilinx family of FPGAs includes Virtex-II and Virtex-4 SoCs, which include PowerPC 405 cores integrated with FPGA logic fabric and high-speed multigigabit transceivers (MGTs). Hybrid architectures like Virtex provide an opportunity to move algorithms between software and hardware state machines readily and are great platforms for experimentation and learning about fundamentals of hardware acceleration and offload. The Xilinx University Program (XUP) Virtex-II includes an embedded software and Electronic Design Automation (EDA) tool chain along with reference designs...
for audio and video processing. See Resources for links.

The first part of this series looked at application-specific integrated circuits (ASICs) for video capture and streaming along with software for basic image processing. Specifically, it examined C code for image edge enhancement, a key step in image segmentation. Image processing is processor-intensive simply because of the large number of operations that must be performed to transform an image pixel by pixel. Using a general-purpose processor to filter, transform, or otherwise process an image can be very inefficient, although software image processing is of course easy to update and debug. A hybrid reconfigurable SoC such as the Xilinx Virtex family, which includes both PowerPC 405 cores and an FPGA fabric, provides a platform on which you can directly compare the trade-offs between processing images in software versus processing them with hardware state machines. For those readers who teach or study embedded systems at the university level, the Xilinx Virtex-II XUP (Xilinx University Program) board offers just such a hybrid reconfigurable platform along with reference designs for audio and video processing (see Resources for a link to Xilinx XUP board, including a reference design for edge detection).

Before going straight to offload with a reconfigurable or configurable SoC, take the C edge enhancement code developed in the previous article in this series and respecify it in SystemC. The SystemC specification allows you to simulate the hardware offload and verify this potential acceleration. The SystemC simulator is freely available and allows readers who don't have access to a hybrid reconfigurable platform like the XUP a chance to consider how well specific image processing algorithms, like edge enhancement, might be offloaded to a hardware state machine.

The point of this example is to consider how future applications will benefit from the rapid migration of software down into hardware state machines in SoCs. This function migration capability of reconfigurable SoCs and families of configurable SoC ASICs distinguishes the SoC as a key technology for emergent applications like Web-based GISes. After finishing the SystemC migration, this article presents a closer examination of these emergent applications, often referred to collectively as Web 2.0. These applications require significantly enhanced resources, both on the server side and the client side, to be fully realized. Furthermore, many exciting embedded applications, like vehicle telematics combined with GIS, can greatly benefit from SoC designs.

**From SystemC simulation to HDL**

Numerous EDA tools are available to convert SystemC specifications to the Verilog hardware description language. They can also convert Verilog code to SystemC, and SystemC code to register transfer language design flow. Likewise, SystemC provides transaction-level modeling and simulation, which make the exploration of function offload a quicker process that can be simulated for early verification, especially for hybrid software and hardware designs.

From C code to SystemC specification

Part 1 of this series examined an edge enhancement example implemented in C code. The code involves the application of a point spread function (PSF; see Resources). The PSF is a 3 x 3 image processing kernel that is applied to every pixel in order to provide a convolution of the original that has sharper edges.

In general, image convolution requires nine multiply and accumulate operations on every pixel in the image, and furthermore on each color subvector (red, green, blue). In our example, this makes for approximately 2 million operations (319 x 239 x 27) at 30 frames per second, which is at least 60 millions of instructions per second (MIPS). Modern embedded processor cores are certainly capable of...
performing many hundreds of MIPS, but recall that the application of the edge enhancement kernel might be just one step in the overall real-time image processing for applications like computer vision or real-time GIS. For example, in the computer vision application system begun in the first part of this series, additional functions might include image segmentation, centroid finding, and tracking.

The application of a PSF kernel to an image is a well-defined algorithm; very little about it is likely to change in a specific implementation, other than the weights used in the PSF (the $K$ value in Listing 1). Given the MIPS required to compute the convolution in software, the clearly defined algorithm, and the potential for speed-up provided by concurrent computation in hardware, the PSF convolution is an ideal candidate for hardware acceleration. Listing 1 provides the SystemC specification of the same edge enhancement image convolution that was provided in C code in the previous installment of this series.

**Listing 1. SystemC specification for simple edge enhancement**

```c
// SystemC implementation
#define K 1

unsigned char R[76800];
unsigned char G[76800];
unsigned char B[76800];
unsigned char convR[76800];
unsigned char convG[76800];
unsigned char convB[76800];


SC_MODULE(psfenhance)
{
    sc_in_clk         CLOCK;
    sc_in<bool>       RESET;
    sc_out<bool>      ERROR;
    sc_out<bool>      READY;

    void compute();

    SC_CTOR(psfenhance)
    {
        SC_THREAD(compute, CLOCK.pos());
        watching(RESET.delayed() == true);
    }
};

void psfenhance::compute()
{
    // reset section
    unsigned i, j;
    bool err;

    while(true)
    {
        // IO cycle for image processing completion
        ERROR.write(err);
        READY.write(true);
    }
}```
The main difference between the original C code implementation and the SystemC specification is the clocked state machine interface that invokes the convolution. The image buffers $R$, $G$, and $B$ are assumed to be populated with data from a video encoder (described in the first part of this series) that has used a DMA (direct memory access) engine to transfer encoded data into the memory buffers. The encoding and DMA interfaces are the same as those used in the first part of this series.

The SystemC specification can be simulated for verification using the SystemC library and simulation.
environment. You can download this environment from OSCI (see Resources). Figure 1 shows a potential design flow for first specifying and verifying the SystemC implementation, then translating it into the Verilog hardware description language, synthesizing it, and finally integrating it for testing on a reconfigurable Virtex-II FPGA platform. The reconfigurable FPGA platform provides a great way to prototype hardware acceleration and offloads before the design is implemented as a custom SoC ASIC.

**Figure 1. Example design flow**

Reconfigurable SoCs and the future of mobile computing

It seems like many of the first-generation mobile computing platforms to date have missed the point. Cell phones and PDAs are often much more of a distraction than an aid in daily life, especially when combined with driving! Future applications should not only improve ubiquity of information availability and access, but should also improve safety, and must provide much more natural real-time interaction with users that does not distract, but rather enhances the human interaction with the world. This has become the goal of the next generation of devices. The quality of service (QoS) and level of user interaction have been relatively low in first-generation devices. The question is, how can the quality be increased while decreasing cost? SoC architectures hold significant promise to help.

Emergent embedded applications (that truly promise to assist rather than distract) like real-time GIS, vehicle telematics, computer vision, voice recognition, and almost any high-QoS form of human-computer interaction and assistance in real time will require significant hardware acceleration of key functions in an SoC design to meet rigorous real-time requirements cost effectively. These applications hold significant promise to improve transportation safety and enjoyment. Location-aware mobile and pervasive computing will be much better integrated with users in vehicles, perhaps even as wearable computers, and provide high-bandwidth interaction in real time; such technology can literally transform how the world is seen.
Configurable and reconfigurable SoCs are critical for the realization of affordable platforms for telematics and a host of pervasive mobile platforms with high-quality, real-time information and processing. At first glance, from a system viewpoint, it doesn't seem to matter how functions and services are implemented as long as they work and perform well. But closer inspection reveals that when a system is evaluated more rigorously in terms of cost and efficiency in providing functions and services, the value of hardware acceleration becomes more apparent. Use of hardware acceleration can decrease power usage, increase performance, and decrease cost. To better evaluate software and hardware acceleration performance, these platforms must provide performance characterized by:

- **QoS**: Latency and jitter in information delivery should be minimized. Software processing is less deterministic than hardware and can introduce jitter.
- **MIPS required per feature**: High-frequency, complex software processing in the main data path can devour MIPS with an insatiable appetite.
- **Cost per real-time stream**: What is the total system cost of delivering a GIS or video stream including both software and hardware?
- **Content storage cost**: What is the cost to store content in flash, on a hard disk drive, or other non-volatile memory device?
- **Content download rates and geographic availability**: When should a mobile system stream and when must it cache data?
- **Power used per real-time stream**: Measuring the number of watts per stream is a good way to gauge the efficiency of a service or stream operation.
- **Platform size, weight, and mass**: Keeping these to a minimum is one of the main reasons to consider embedded SoC designs and hardware acceleration.
- **Service availability and reliability**: Global mobile platforms for telematic applications will not provide perfect operation all the time, but must be available and reliable enough to win consumer trust. Mean time between failures should be kept high and mean time to recovery kept low.

In essence, these future applications must be able to operate in real time with reliability that meets or exceeds human performance so that ultimately these platforms can be relied upon to offload and assist human perception and decision making. This will require levels of performance, embedding, and SoC integration levels that are unprecedented, but promise many benefits by amplifying our ability to perceive and operate in an ever more complicated world. Real-time human-computer interaction and QoS that have previously only been found in exotic military and research systems promise to show up in a car near you soon. One has to wonder: will the mantra "hang up and drive" someday turn into "please plug in before you drive"?

Resources

**Learn**

- "SoC design for hardware acceleration, Part 1" (developerWorks, June 2006): The first part of this series includes the software implementation of our algorithm along with more background on video streaming and image processing applications and hardware engines.

- Find definitions for the basic terminology used in this article related to image processing and digital signal processing on Wikipedia:
  - **PSF** (point spread function)
  - **Convolution**
  - **FIR** (finite impulse response)
  - **GIS** (geographical information systems)
Vehicle telematics

- **The Scientist's and Engineer's Guide to Digital Signal Processing**: An excellent guide for those new to DSP and image processing.

- Many interesting issues arise as in-vehicle telematics continue to be developed. The following links provide information on the research being conducted by IBM:
  - **Automotive Telematics Data Privacy Protection Framework**: Details of safety and data privacy
  - **IBM delivers speech recognition navigation system for Honda in 2005**: Work on hands-free navigation
  - **IBM Mobile and Pervasive Computing project**: More fundamental research

Get products and technologies

- **OSCI**: Download the SystemC library and simulation environment.

- **The Xilinx University Program Virtex-II platform**: Provides an excellent reconfigurable SoC for exploring audio and video processing hardware acceleration and offload.

- **Xilinx University Program Virtex-II platform reference designs**: For DSP, audio, and video processing including a reference design for edge detection. The AC97 audio codec is built into the XUP, making this a great platform for exploring offloads for applications like Voice over Internet Protocol.

- **Digilent VDEC1 board with Analog Devices ADV7183B video decoder**: You can add this board to the XUP to make it a great platform for video streaming and image processing hardware acceleration work.

- **Digilent**: Purchase the XUP board and accessories here.

- The race is on to provide the ultimate Web-based GIS mapping systems:
  - **Google Earth**: The latest version has a new GIS Web viewer.
  - **Virtual Earth**: Microsoft's similar Web-based application
  - **Yahoo Local Maps Beta**: Yahoo's GIS API that is used to enable this site

- The race to build the ultimate in vehicle telematics platform is on as well:
  - **AUTOSAR**: Standards for vehicle platforms are being defined by groups like this.
  - **IBM Automotive**: IBM's automotive and telematics solutions Web site
  - **Wind River Platform for Automotive**: Embedding software and hardware support with SoCs is a key for building these systems.

About the author
Dr. Sam Siewert is an embedded system design and firmware engineer who has worked in the aerospace, telecommunications, and storage industries. He also teaches at the University of Colorado at Boulder part-time in the Embedded Systems Certification Program, which he co-founded. His research interests include autonomic computing, firmware/hardware co-design, microprocessor/SoC architecture, and embedded real-time systems.

Trademarks