Building real-time HDTV applications in FPGAs using processors, AXI interfaces and High Level Synthesis Tools

Kees Vissers, Stephen Neuendorffer
Xilinx
2100 Logic Drive
San Jose, CA 95124, USA
kees.vissers@xilinx.com,
stephen.neuendorffer@xilinx.com

Juanjo Noguera
Xilinx
One Logic Drive
Citywest Business Campus
Saggart, County Dublin, Ireland
Juanjo.Noguera@xilinx.com

***Abstract*—Modern FPGAs enable complete system designs that include processors, interconnect systems, memory subsystems and a number of application functions that are implemented using High-Level Synthesis tools.**

Keywords-HDTV systems, processor subsystem, image processing applications, High-Level Synthesis.

I. PROCESSING HDTV STREAMS IN FPGAS.

Modern FPGAs consist of high-speed I/O, conventional LUTs, and many distributed small memories. The latest silicon technology for FPGAs makes it possible and affordable to build complete HDTV processing systems in a consumer priced solution. The systems often consist of hard processors or soft processors with a modern interconnect and a number of dedicated processing blocks. An illustration is given in Figure 1.

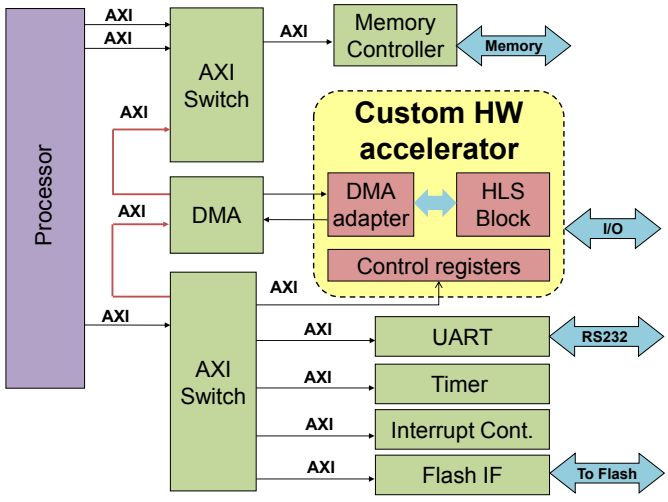

Figure 1: A system architecture with processor and accelerators in FPGAs

These systems contain a number of accelerators, dedicated memory interfaces and I/O to the external system. Many modern video SoCs have a similar architecture, integrating fixed-function video processing blocks with a small amount of customizable processing implemented in programmable processor cores. The key difference is that FPGAs can enable custom, high performance pixel processing algorithms to be implemented in an HDTV system. When combined with state-of-the-art High-Level Synthesis (HLS) and predefined system templates defining standard video interfaces, FPGAs become a unified platform for video applications which can be completely programmed in C/C++ by an end-user.

However, implementing this flow requires several significant pieces of technology. Automatic and optimized High-Level Synthesis for the FPGA fabric is needed to program accelerators. A system architect must be able to engineer and verify the system template and guarantee Quality of Service requirements to the end user without knowing the exact function that will be implemented. An end user must be able to understand the performance and resource usage of a C/C++ program partitioned between multiple processors and FPGA accelerators without getting stuck in the minutiae of the system architecture. We believe that the fundamental technology to solve these challenges exists today.

II. HIGH – LEVEL SYNTHESIS TOOLS

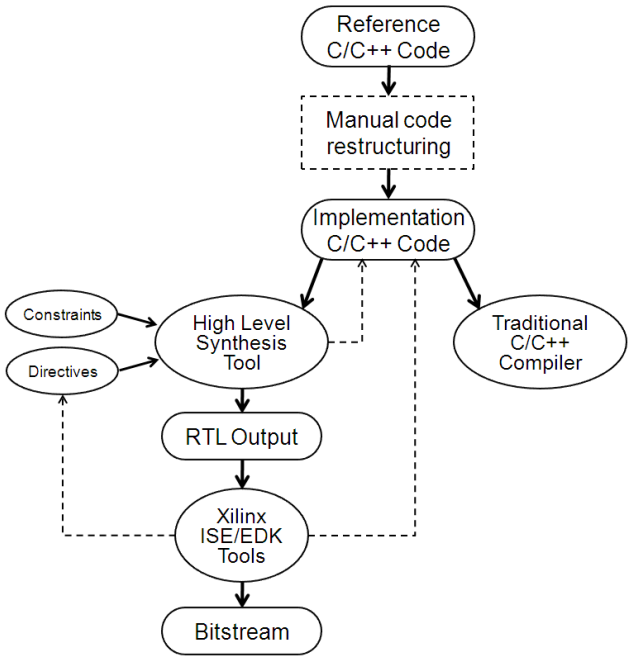

Figure 2: High-level Synthesis for FPGAs

978-3-9810801-7-9/DATE11/©2011 EDAA

High-level synthesis tools take as their input a high-level description of the specific algorithm to implement and generate the RTL description of an FPGA implementation. Modern high-level synthesis tools accept *untimed* C/C++ descriptions as input specifications. These tools give two interpretations to the same C/C++ code: (1) sequential semantics for input/output behavior; and (2) architecture specification based on C/C++ code and compiler directives. Based on the C/C++ code, compiler directives and target throughput requirements, these tools can predictably generate high-performance pipelined architectures that are often as good as can be written manually in RTL, in a shorter amount of time. [1] Among other features, high-level synthesis tools enable automatic pipeline stages insertion and resource sharing to reduce FPGA resource utilization. In summary, high-level synthesis tools raise the level of abstraction for FPGA design, and make transparent the time-consuming and error-prone RTL design tasks. The overall design approach for the creation of an accelerator block is shown in Figure 2. The design of an accelerator now becomes and iterative design process all in C/C++ as shown in Figure 3.

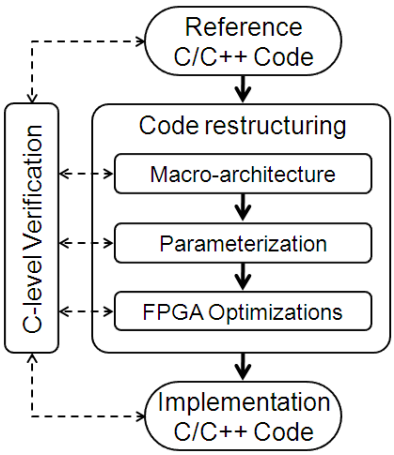

Figure 3: Iterative C/C++ refinement design approach

III. HDTV PROCESSING

In modern HDTVs the size of the image is often 1920 by 1080 pixels. When a progressive format is used, which broadcasts all the video lines in the frame, this is often referred to as a 1080p format. Commercial frame refresh rates range from 60 to 240 frames per second or more. The high frames rates are often used to reduce motion artifacts. Pixel data is often represented in either a luminance format, e.g. YUV, or in a color format, e.g. RGB. The number of bits used to represent a pixel is in the range of 24 to 36. When using 32 bits per pixel and 60 frames per second on an image size of 1920 by 1080 pixels this gives a continuous video stream of 1920*1080*60*32 = ~ 4 Gbit/s or ~ 500 Mbyte/s. In a typical video processing system there will be several streams to an external memory, or video frame buffers, and several video input and output streams. It is clear that these systems need to be designed in such a way that no pixel data gets lost, while keeping the buffers in the system as small as possible in order to reduce cost.

For pixel processing accelerators, these data rates can be hard to accomplish. With adequate buffering, the average pixel rate is approximately 125 Msamples/second. For a processor, this typically means that a very small number of instructions can be issued per pixel, even when making use of SIMD parallelism. In FPGA fabric, on the other hand, it is common to implement heavily pipelined accelerator architectures at 125 MHz, even in slow consumer-grade FPGAs. Furthermore, for many algorithms, these frequencies can be reached from High-Level Synthesis. As a result, even pixel-processing algorithms requiring hundreds or thousands of operations per pixel can be easily implemented. [2]

However, more difficulty arises when the accelerators must be integrated in a system communicating with external memory. Guaranteeing the system performance of these streams, when combined with other dynamic memory traffic can be a significant design problem. The system processors need to setup the rapid transfer of large amounts of data to and from external memory frame buffers, while keeping up with the real-time data requirements of the external video interfaces. This architecture implies significant performance requirements on the DMA subsystems, the processor response time and the interconnect response times. Furthermore the interface to external memory needs to be designed so that it can handle the worse case workload of all video streams and memory references from the processors. This is an important system level consideration.

To understand these system-level considerations, we consider a system with one external HDTV video input, one external HDTV video output and 2 video accelerators each with two input streams and one output stream. In order to decouple the interfaces from the accelerator processing, we assume that the input is written directly to the external memory and the output is read directly from external memory. We also assume that each accelerator reads two inputs from external memory and writes a single output to external memory. For simplicity we assume all references are a single sequential stream. In this system we now have 3 video streams writing to external memory (the video input, and the two outputs of the accelerators), and 5 video streams reading from memory (two inputs per accelerator, and one output stream). The minimum bandwidth that needs to be supported for these video streams is then 8 * 500MByte/s = 4 GByte/s. Furthermore a typical control processor will generate a number of requests to external memory based on the transfer of data from the Instruction Cache and Data Cache. When you take a buffer of 4Kbyte at each input and each output in the system, this allows to buffer 1000 samples (4 byte per sample). A memory controller will have an arbitration algorithm that needs to arbitrate between video streams and low latency requirements for cache misses. Typical cache miss latency needs to be in the range of 10 to 100 cycles, where typical video streams can tolerate high latency but have stringent throughput requirements, as mentioned above. A typical system has to be designed to support the bandwidth and real-time requirements of all the streams. The easiest way to guarantee these requirements is probably to include a Quality of Service mechanism in the memory and interconnect subsystem. In general, the challenge

is to dimension the buffers and memory system small enough while guaranteeing system level performance. Typical numbers are that accelerators are stalled due to the interface (buffers empty or full, due to interconnect or memory controller) no more the 25% of their total cycles.

In many systems including in modern HDTV products the total system can consist of a combination of dedicated existing SoCs and some FPGA technology. In modern security systems the use of FPGAs is becoming an important implementation technology to achieve programmable HDTV video systems. These systems often combine a network interface, e.g. 100Mbs or 1Gbs Ethernet interface and a camera or display interface. The end-to-end system performance requirement often requires a system analysis and performance simulation environment. In FPGA systems, the rapid implementation of the first prototype of the final product provides an early route to implementation and experimentation on the FPGA based system itself that is often not possible in the existing SoC development cycle.

IV. Conclusion

FPGAs are well positioned for use in high-performance video processing systems. When combined with High-level Synthesis and video platforms with guaranteed quality of service, they can also be very easy to program.

V. References

[1] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-Level Synthesis for FPGAs: From Prototyping to Deployment", IEEE Transactions on CAD, to appear.

[2] K. Denolf, S. Neuendorffer, and K. Vissers, "Using C-to-gates to program streaming image processing kernels efficiently on FPGAs", Proceedings of the conference on Field Programmable Logic (FPL), Sep. 2009.