

Streams-Based Programming Accelerates FPGA-Based Embedded Applications

You can simplify the acceleration of complex hardware/software applications with streams-oriented C-language programming methods.

by David Pellerin
CTO

Impulse Accelerated Technologies, Inc.
david.pellerin@impulsec.com

The Xilinx® Virtex™-4 FX family of devices offers high-performance embedded systems developers a flexible, powerful, and highly customizable solution for hardware-accelerated embedded applications. Recent advances in software-to-hardware compilers have made it easier for software application developers to move computationally intensive algorithms into dedicated hardware accelerators. For software programmers accustomed to writing applications for traditional processor targets, however, it may not be obvious how to take full advantage of parallelism when targeting an FPGA.

In this article, I'll describe the key areas in which software-to-hardware tools can help you manage the complexity of mixed hardware/software applications and describe a C-language programming model that is ideal for FPGA-based hardware-accelerated applications.

C-Language Parallelism

To some, the idea of describing parallel hardware applications using the C programming language may seem counterintuitive. Indeed, standard ANSI C represents a relatively simple and inherently sequential programming language that does not naturally lend itself to parallel algorithms. You cannot use standard C to describe parallel systems or generate parallel computations unless the language has been extended in some way to support explicit parallelism, or unless you

have access to specialized compilers that can extract low-level parallelism from otherwise sequential C statements.

Fortunately, for system- or application-level parallelism there are extensions to C that support various parallel programming models. The most widely recognized of these extensions is threading. The standard C language does not include support for threads, but many thousands of software programmers today use C-language thread libraries (such as POSIX threads or the Microsoft Windows threading library) to manage multiple parallel computations and to describe mutexes, condition variables, and other elements required in a threaded application.

Another less common C library for parallel processing is the message passing interface, or MPI. This programming

Impulse C simplifies the creation of highly parallel algorithms – including mixed software/hardware algorithms – through the use of well-defined data communication, message passing, and synchronization mechanisms.

library is primarily used for targeting parallel computing clusters and high-end supercomputers. MPI allows C programmers to describe a computationally intensive application as a collection of connected processes using a message-passing method of communication and synchronization. The MPI programming model allows a software process running on one node of a computing cluster to pass messages and coordinate its operation with software processes running on

other nodes in the system. MPI applications may involve different algorithms and computations running on each node, or involve parallel arrays of identical software processes, each operating on different sets of data.

The Impulse C Streaming Programming Model

Impulse C is a parallel programming function library and compiler based on standard ANSI C. Impulse C extends the C language in support of parallel programming while remaining compatible with standard C tools such as debuggers and profilers. Impulse C simplifies the creation

of highly parallel algorithms – including mixed software/hardware algorithms – through the use of well-defined data communication, message passing, and synchronization mechanisms.

Figure 1 illustrates how you can partition such an application to create a multiple-process implementation of a complex image analysis. These individual processes can then be implemented as communicating hardware and software components or implemented entirely within an FPGA. In this example, a collection of parallel, pipelined processes perform an image processing operation. Each process has been described using one or more C subroutines.

On the software side of the application – for example, on an embedded PowerPC™ 405 processor – Impulse C function calls are used to open and close data streams, read or write data on the streams, and check for stream status. You can use related Impulse C functions to send process-to-process messages or poll for results. For the Virtex-4 FX device, you can specify stream reads and writes as operations that take advantage the auxiliary peripheral unit (APU) interface, which provides a high-performance software-to-hardware serial interface capable of transferring as many as 128 bits of data in a single transaction.

Impulse C is designed for dataflow-oriented streaming applications, but it is also flexible enough to support alternate programming models, including the use of shared memory as a communication mechanism. The programming model that you, as an Impulse C programmer, will select will depend not only on the requirements of your application but also on the architectural constraints of your programmable platform target. For some targets, including the Virtex-4 FX FPGA, data streaming provides a high-throughput method of communication between the processor and the FPGA accelerator. In other platforms, a shared memory approach allowing burst transfers of data may be more appropriate.

At the heart of the Impulse C streaming programming model are processes and streams. Processes are independently synchronized, concurrently executing seg-

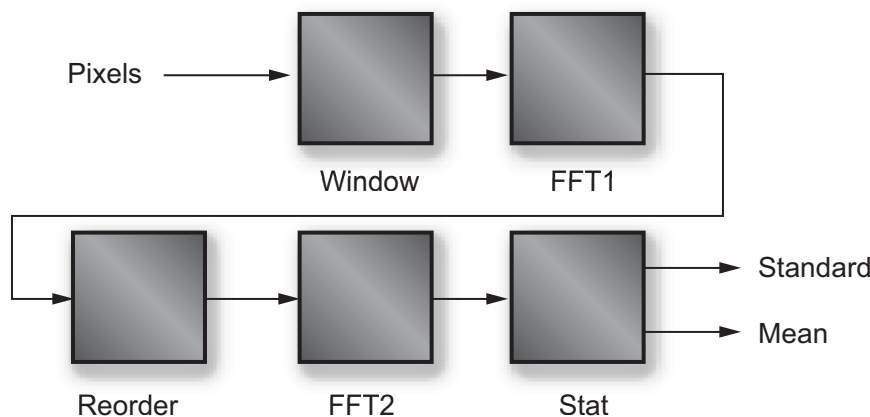


Figure 1 – A complex image filter application developed using a streaming programming model.

other nodes in the system. MPI applications may involve different algorithms and computations running on each node, or involve parallel arrays of identical software processes, each operating on different sets of data.

Streaming is another programming model with characteristics similar to MPI, allowing for data communication and synchronization between multiple related processes. Streaming is an ideal programming model for many applications in the domain of high-performance embedded computing (HPEC), in which low-latency yet complex data computation is required. As you will see, the streaming program-

ming model is ideally suited to mixed processor/FPGA platforms such as the Virtex-4 FX device.

In Impulse C streaming applications, hardware and software processes communicate primarily through buffered data streams that are implemented directly in hardware. This buffering of data, which is implemented using dual-clock FIFO interfaces generated by the compiler, makes it possible to write parallel applications at a relatively high level of abstraction without the cycle-by-cycle synchronization that would otherwise be required.

ments of your application. A subset of standard C is used to write hardware processes and perform the work of your application by accepting data, performing computations, and generating outputs. Streams provide a buffered, self-synchronizing method of managing data communications between the processes.

In a typical application like the pipelined image filter described earlier, the data processed by your application will flow from process to process by means of streams, or in some cases by means of messages or shared memories. You can specify the characteristics of each stream, including the width and depth of

- Stream data packet sizes are a fixed size typically ranging from 8 to 128 bits.
- Multiple related but independent computations must be performed on the same datastream. This characteristic allows the algorithm to be parallelized through the use of a system-level pipeline, or by splitting the data and creating an array of connected parallel processes.
- References to local or shared memories, usually expressed in the form of C arrays, which may be used for storage of data, coefficients, and other constants or to deposit the results of computations.

sequential C-language statements. The compiler must also evaluate structures such as loops at a high level and use techniques such as unrolling, pipelining, and instruction scheduling to increase throughput. Branch structures, assignments, and intermediate operations must be analyzed and, where possible, executed in parallel. The Impulse C compiler is capable of such optimizations and can save you a great deal of time. Related Impulse tools make it possible to interactively analyze the generated parallel structure in support of iterative optimizations.

A second and equally important function of such a compiler is to generate the necessary process-to-process interfaces, whether those interfaces are implemented as direct hardware connections, as FIFO buffers or, in the case of hardware-to-processor interfaces, as standard buses or platform-specific serial or DMA connections.

The Impulse C compiler performs both of these important functions and generates hardware in either VHDL or Verilog. You can then synthesize the FPGA hardware using tools such as Xilinx Platform Studio or ISE™ software. On the processor side, the compiler generates run-time libraries ready for use on the embedded PowerPC or MicroBlaze™ processor, or on an external processor in the case of supported board-level platforms.

Conclusion

Streams-based programming provides an alternative way for developers of high-performance embedded systems to create mixed software/hardware applications. The streaming programming model used by Impulse C is ideal for FPGA-based platforms, particularly when those platforms include high-bandwidth interfaces between an embedded processor and dedicated coprocessors implemented in the adjacent FPGA fabric. The Virtex-4 FX FPGA provides one such platform and is a perfect target for hardware acceleration of complex embedded software applications.

For additional information about streams-oriented programming and C-to-FPGA compilation, visit www.impulsec.com/xilinx.

```
co_stream_open(sample_stream, O_RDONLY, INT_TYPE(32));
co_stream_open(result_stream, O_WRONLY, INT_TYPE(32));
do {
    co_stream_read(sample_stream,&sr,sizeof(int32));

    ... // Process the data here

    co_stream_write(result_stream,&si,sizeof(int32));
} while (. . .);
```

Figure 2 – You can use C library functions to describe streaming operations.

the generated FIFOs, in the C application. Actual stream operations are described using C functions such as those illustrated in Figure 2.

A Wide Range of Applications

You can express an unlimited number of applications using a streaming programming model, but the most efficient applications are those that have some or all of the following characteristics:

- High data rates to and from data sources and between processing elements. The source of the data may be a processor such as an embedded PowerPC, or it may be a direct hardware interface such as a camera or other data-acquisition device.
- Multiple independent processes communicating primarily through the data being passed, with occasional synchronization requested through messages.

Impulse C is specifically designed to address such applications, which are found in a wide variety of domains including:

- Signal and image processing
- Voice and data communications
- Data compression and encryption
- Scientific computing
- Bioinformatics
- Financial modeling

Generating Parallel FPGA Hardware

When generating a hardware/software system or a multi-process, streaming application written in C, the compiler must perform two important functions. First, the compiler must be capable of analyzing each of the parallel processes (which may each be represented by a single C subroutine or by a collection of subroutines) and extract low-level parallelism from otherwise