

USING C-TO-HARDWARE ACCELERATION IN FPGAS FOR WAVEFORM BASEBAND PROCESSING

David Lau (Altera Corporation, San Jose, CA, dlau@altera.com)
Jarrod Blackburn, (Altera Corporation, San Jose, CA, jblackbu@altera.com)
Charlie Jenkins (Altera Corporation, San Jose, CA, chjenkin@altera.com)

ABSTRACT

Software-defined radio (SDR) architectures typically include general-purpose CPUs (GPPs), digital signal processing (DSP) ASSPs and FPGAs that process different waveforms, functions, and algorithms. GPPs typically handle network protocol processing and management functions. Historically, DSPs handled transceiver baseband processing and encoding, while FPGAs provided high-performance IF up/down conversion and preconditioning functions. Now FPGAs, when used with embedded soft-core processors, have absorbed the DSP baseband processing and some GPP functionality as well, providing a smaller, lower power solution. However, meeting the baseband performance requirements requires aggressive use of hardware acceleration. In this paper, we discuss an efficient methodology for hardware acceleration of SDR waveforms, the creation and use of hardware acceleration units, and a tool that automates the flow. The Altera® Nios® II C-to-Hardware (C2H) Acceleration Compiler is a coprocessor generation tool that converts performance-critical ANSI C functions into hardware accelerator modules with direct memory access. Results are presented showing performance gains of 13–73X over software only, offering a promising solution for rapid development of high-performance SDR systems.

1. INTRODUCTION

In the past, FPGAs were used as a convenient interconnect layer between chips in a system. In SDRs, FPGAs are now being used as programmable up/down intermediate frequency and signal processing hardware that boost performance while providing lower cost and lower power. Typical implementations of SDR modems include a GPP, DSP, and FPGA. Today's latest generation FPGAs can also be used to offload the GPP or DSP with application-specific hardware acceleration units. Soft-core microprocessors can easily extend their functionality with custom logic and hardware acceleration coprocessors added to the system. Furthermore, by using general-purpose routing resources

available in the FPGA, these hardware acceleration units can run in parallel to further enhance the total computational throughput of the system.

Three different methods for hardware acceleration of SDR waveforms have been previously discussed at this conference: custom instructions, arithmetic coprocessing units, and application-specific instruction-set processors [3]. In this paper, we will focus on arithmetic coprocessing units and the automated design flow made possible by Altera's Nios II C2H Compiler. This compiler provides a pure-software design flow, automatically moving user-specified performance-critical functions from software running on the FPGA processor into pipelined, optimized hardware accelerators. These accelerators have direct access to the processor's memory subsystem and can sustain extremely high bandwidth through parallel transactions to an arbitrary number of buffers.

Sections 2 and 3 provide background information on SDR and system architecture. An efficient methodology for developing hardware coprocessors using a slave-side arbitration interconnect is discussed in Sections 4 and 5. In Sections 6 and 7, the automation of this flow with the Nios II C2H Compiler is discussed, followed by optimization strategies in Section 8, user test results in Section 9 and the summary in Section 10.

2. SOFTWARE-DEFINED RADIO

The concept behind SDR is that more waveform processing can be implemented in reprogrammable digital hardware so a single platform can be used for multiple waveforms. With the proliferation of wireless standards, future wireless devices will need to support multiple air interfaces and modulation formats. SDR technology enables such functionality in wireless devices by using a reconfigurable hardware platform across multiple standards.

SDR is the underlying technology behind the Joint Tactical Radio System (JTRS) initiative to develop software-programmable radios that enable seamless, real-time communication across the U.S. military services, and with coalition forces and allies. The functionality and expandability of the JTRS is built upon an open architecture

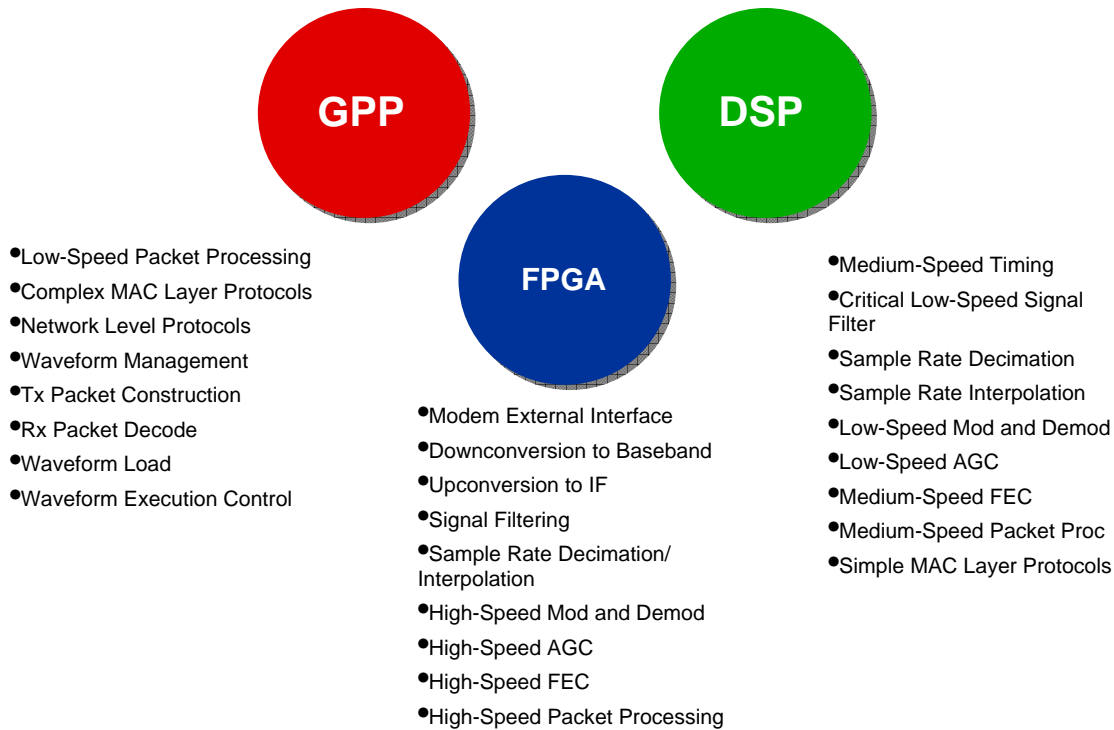


Figure 1. Example Architecture Splitting SDR Functions Across GPP, DSP, and FPGA

framework called the software communications architecture. The JTRS terminals must support dynamic loading of one or more of 25 specified air interfaces or waveforms, typically more complex than those used in the civilian sector. To achieve all these requirements in a reasonable form factor requires extensive, yet different processing powers.

3. SDR SYSTEM ARCHITECTURE

Most SDR systems utilize GPP, DSP, and FPGA in their architectures. These general-purpose processing resources can be used for different parts of the overall SDR system, and Figure 1 shows the typical functions found in an SDR divided across each of these devices. However, there is a significant amount of overlap between each of these elements. For example, an algorithm running on the DSP could be implemented in the GPP, albeit more slowly, or rewritten in HDL and run much faster in an FPGA as a coprocessor or hardware acceleration unit.

4. HARDWARE COPROCESSORS

Typical system design flow begins with a pure-software specification running on a CPU, then uses profiling to determine algorithmic bottlenecks. These bottlenecks can be reduced in several ways, including the creation of custom hardware accelerators that offload the CPU, exploiting

parallelism in the algorithm to achieve significant performance increases. This method works best when dealing with complex computation-intensive algorithms that operate on large blocks of data. Unlike custom instructions, hardware accelerators are autonomous—once activated, they can run without intervention from the CPU, creating thread-level parallelism as well as instruction-level parallelism.

Accelerator modules contain data master ports that connect directly to the CPU's memory subsystem. With an FPGA-based interconnect (such as Altera's Avalon[®] memory mapped and streaming interfaces), it is possible to enable an arbitrary number of simultaneous transfers by arbitrating between multiple masters on the slave side. Accelerator modules can then be created with multiple master ports that simultaneously access different memory buffers, allowing much higher memory bandwidth than that achievable by a CPU with a single data master.

5. SYSTEM INTERCONNECT

The FPGA-based Avalon system interconnect fabric is automatically generated by Altera's SOPC Builder system integration tool. Similar to a fully switched system bus in functionality, it uses dynamically generated switches and wires to interconnect modules. These can be soft intellectual property (IP) blocks, custom user logic, or interfaces to off-chip peripherals. This approach overcomes the bottleneck

