Architecture Brief

Altera SoC Software Development Tool Flows

Introduction

SoC silicon with an integrated FPGA element adds end-device functionality, but can add complexity for processor software developers. Separate design flows for processor and FPGA development allow developers to remain in the existing development environment using appropriate and familiar development tools.

This Architecture Brief describes the software development tool flows to maximize development.

Key aspects of this Architecture Brief are highlighted in an online video: "Nearly everyone on the planet knows".

To provide the preferred design flows for SoC FPGAs to both embedded software developers and FPGA designers, four key areas, software debug, hardware/software handoff, boot sequence and OS support, need to be addressed. It follows that the software developer is able to use the same tools, infrastructure and methodology as with any other SoC. Any functionality required due to the integration of the FPGA, is kept as separate as possible. This allows design flows where separate processor and FPGA development teams continue to develop as if the integrated SoC-FPGA components are separate devices.

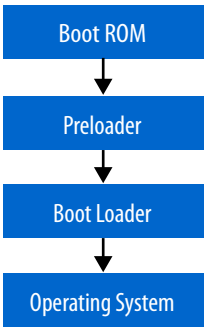
However, there are benefits to the processor software developer in having the FPGA integrated with the processor, particularly in driver development and debug tools. Debug tools treat the FPGA design as if it is a hardened integrated ASIC component. This removes the FPGA complexity, allowing development and debug without the need to learn the FPGA adaptive debug and debug and trace tools. (See also the Architecture Brief – FPGA-Adaptive Debug, in this series.)

By importing generated files that defined the hardware from the Altera Qsys and Quartus II FPGA design tools into the embedded software design flow, complete separation of design environments is maintained while still leveraging the benefit of processor and FPGA integration.

Hardware-Software Hand-off and Preloader

The processor boot sequence is illustrated in ***Figure 1***.

Figure 1: Processor Boot Sequence

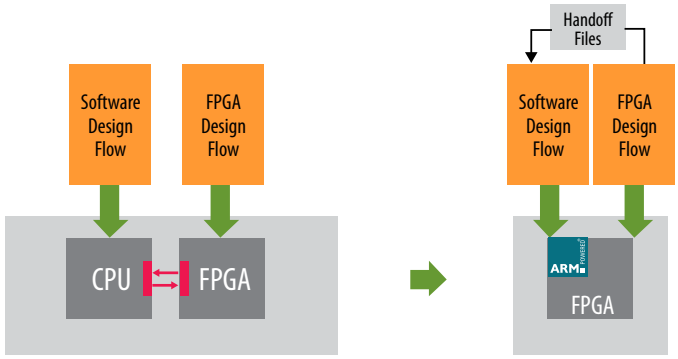


This modularity gives maximum flexibility to the software developer programming the processor. The boot ROM is fixed and loads the preloader from the location as defined by the boot select pins. The preloader is auto-generated from the FPGA design tools, and is the first code run from the boot source. Its purpose is to initialize the programmable elements of the silicon that have FPGA dependencies like pins and SDRAM hardware configuration, and to place the silicon into a known state for the bootloader. The chosen bootloader is then configured and run like any other SoC bootloader. The bootloader may be the final image the Hard Processor System (HPS) runs, which is normally the case if bare-metal or a very thin RTOS is used, or may load and run a High Level Operating System (HLOS) such as Linux. The separation of the preloader and bootloader allows for all configurable elements of the silicon due to the FPGA to be automatically generated and configured without causing non-standard dependencies if a standard bootloader is used (such as uboot).

Software Debug Tools

Separate processor and FPGA design flows extends to the debug tools as well. This is illustrated in ***Figure 2***.

Figure 2: Debug tool options for processor and FPGA design flows



The SoC on the right features a standard ARM processor and subsystem; the entire ARM ecosystem is available to leverage in choosing a debug toolchain. A separate JTAG header is available for both the processor and FPGA domains, which allows for no dependency between the two.

The FPGA is programmable and therefore its hardware is not static from the processor viewpoint . To allow the user to inspect and change the user created hardware registers in the FPGA as if it were fixed hardware, Altera and ARM jointly developed the ARM® Development Studio 5 (DS-5™) Altera Edition Toolkit as the first FPGA-adaptive, unified debugger for Altera SoCs. FPGA-adaptive debug is a means for the debugger to see and export any user-created hardware registers in the FPGA to the processor debug view. It removes the debug barrier between the processor subsystem and the on-chip FPGA for visible control. The means to provide this is in a hand-off file created by the FPGA toolchain. When a new FPGA bitstream is created, a separate .svd file is generated that may be imported into the DS-5 debugger that gives the register view of the bitstream hardware. With the correct .svd file, the processor software developer is able to create and debug drivers and other low-level software for the FPGA image as if it were static hardware.

Linux

Linux development occurs in the same fashion as any other SoC. A .git tree is cloned (https://www.rocketboards.org/foswiki/Documentation/GitWeb) and the kernel built. The kernel may be built independently or as part of a larger build system (e.g. Yocto) or distribution (e.g. the Angstrom distribution). There is no required dependency on the FPGA design tools.

The Linux kernel, particularly for embedded systems, takes advantage of the device tree features to make managing integrated peripherals easier. To seamlessly integrate the FPGA image into the kernel Device Tree, the Quartus II FPGA design tool exports a System in a Programmable Chip (SOPC) Information (.sopcinfo) file when a bitstream is created. This file may be used by the device tree generator to create the device tree used by the Linux kernel.

Partner Operating Systems

A plethora of operating systems, each with specific Board Support Packages (BSPs), have been ported to various Altera SoC FPGA devices. Each of these OSes boot and operate as if on any other SoC. If the OS supports device trees (e.g. VxWorks), then the SOPC information file that is generated when a bitstream is created may be used in the device tree generator to create the device tree for the FPGA image. If device trees are not used, then the specific OS driver must be integrated by hand in whatever driver infrastructure the specific OS uses.

Conclusion

Separating the processor and FPGA tool flows provides familiarity for both software developers and FPGA designers. It allows them to use their preferred design flows for SoC FPGAs. Enhancements made by Altera to its DS-5 debugger from ARM allow users to see and export FPGA registers, in the processor debug view, allowing maximum flexibility for the software developer without having to learn FPGA tools. There is also a broad range of OS support for Linux and other partner OSes, to manage integration in the SoC FPGA without depending on FPGA tools.

Design Verification Tools Category
DesignCon 2014

Altera Corporation
101 Innovation Drive
San Jose, CA 95134
USA
www.altera.com

Altera European Headquarters
Holmers Farm Way
High Wycombe
Buckinghamshire
HP12 4XF
United Kingdom
Telephone: (44) 1494 602000

Altera Japan Ltd.
Shinjuku i-Land Tower 32F
6-5-1, Nishi-Shinjuku
Shinjuku-ku, Tokyo 163-1332
Japan
Telephone: (81) 3 3340 9480
www.altera.co.jp

Altera International Ltd.
Unit 11- 18, 9/F
Millennium City 1, Tower 1
388 Kwun Tong Road
Kwun Tong
Kowloon, Hong Kong
Telephone: (852) 2 945 7000
www.altera.com.cn