

Architecture Brief

Altera SoC FPGA-Adaptive Debug

Introduction

By integrating processor, peripherals and FPGA into a single chip, SoC FPGAs can make faster, less expensive and more energy-efficient products possible. This innovation in hardware, however, must be matched with support in software development and debug tools to bring these features to life.

In their 2014 Embedded Market Study, UBM and EETIMES found that "*Meeting schedules* remains the premier challenge for development with the *debugging process*... not far behind." Indeed, the debugging process can present one of the longest development phases. Any tool that speeds up debug in order to meet schedules is of high value. One challenging phase in the debug process is that of optimizing performance and power, without ripping apart ones' design.

This Architecture Brief describes how the Altera SoC Embedded Design Suite (EDS) toolset, which includes the ARM[®] Development Studio 5 (DS-5[∞]) Altera Edition Toolkit, can be used to quickly and confidently debug an Altera SoC FPGA.

At the center is FPGA-Adaptive Debug capability.

Development Tool Challenges

Programmability of the FPGA means that engineers might re-program the hardware during the course of the development project; hardware can even be re-configured during runtime. This has two important software implications:

- 1. The CPU software and the FPGA programs must be developed and debugged alongside each other; in a traditional SoC, the embedded software is developed on top of fixed hardware.
- 2. The FPGA hardware definition is user-defined, therefore the software development tools and board support packages (BSPs) that ship with the SoC FPGAs will support all the standard peripherals, but they are not pre-loaded with any memory map information or debugging hooks for the FPGA-based peripherals the hardware team may create.

ARM DS-5 Altera Edition Tool Kit

To support the unique advantages and features of Altera SoC FPGAs without requiring a new set of vendorproprietary tools, Altera teamed with industry leader ARM to develop a special edition of the industry-standard ARM DS-5[™] Toolkit to support Altera SoC FPGAs. The ARM DS-5 Altera Edition Toolkit, offers FPGA-adaptive debug and other key multicore features using the familiar, industry-standard ARM DS-5 interface (*Figure 1*). The package also enables the use of a single Altera USB-Blaster[™] II cable for both hardware and software debug.

Figure 1: The ARM DS-5™ Altera Edition Toolkit Interface is Already Familiar to Many ARM Developers

le Edit Source Refactor Navigate Search Project B	in Window H								_		
	an Munaam D	elp									
📬 र 🗟 🕲 🚔 🖗 र 🚨 🥭 🖉 र 🧕	9 - 6 -	\$\$ \$\$ • \$\$ •						2 4]		
Debug Cont 🕴 💫 Project Expl 📕 Remote Sys 🧮	Comman	ds 🕄 📕 History 🧐 Scripts	- 5	9	Breakpoints 👯 Express	ions (R) Func	tions 23		400		
	7	8	B 58 - 4 -	2		1 Linked: Alt	era SoC FPGA	-0.			
□ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	•	Linked: Altera SoC FPGA -				Name			Start Address		
R Altera SoC FPGA connected	wait		. 0	altera_gpio_get		\$dx7F00000 - \$dx7F000038 \$dx7F000068 \$dx7F00068 \$dx7F0009C					
Contractive Threads	Execution	00018		altera_gpio_set altera_gpio_dir_in							
So blinky #4 stopped on breakpoint #2 (PID 481 w	In thread			altera_gpio_dir_out							
arch_local_ing_save S:0x7F000038 85,0 {					e altera gpio save regs				S:0x7F0000F0 -		
altera_gpio_set=0x8	continue		1	* [m				,			
gpio_set_value_cantleep+0x6C gpio_value_store+0xC0		Execution stopped at breakpoint 2: 5:0x7F000			Variables 23			AVE			
dev_attr_store=0x18	In thread S:0x7E000			Variables 1-1	de tinhad Ale	Altera SoC FPGA •					
flush write buffer+0x40	wait	S:0x7F000038 85,0 { wait			Name	Value	Type		nt Size Locatio		
sysfs_write_file+0:d08	continue			R	Cocais	1 variable	type	Cour	ic Size Locatio		
vfs_write+0x44		Execution stopped at breakpoint 2: 5:0x7F000 In thread 4 (05 thread id 481)			- o flags		27 long unsign		ed int 32 \$R3		
sys_write+0x40	_ 5:0x7F000	038 85,0 {			File Statics (current)						
			*		🗁 Globals	954 variables					
Altera SoC FPGA connected	Command:	Press (Ctrl+Space) for Content Assist	Submit	I e		m			_		
79 * §val: Value to be written to specified 0 * 51 * This function writes the specified value 2 * 6010 evice. 53 */ 54 static void altera_gplo_set(struct gplo_chip 55 unsigned long flags;	in to the	Address Opcode Disass S:0x77000024 SS000 LDRNE S:0x77000024 SS000 LDR S:0x77000024 SS000 LDR	r3,[r0,#0x50] r3,[r3,#0] r1,r3,r1 r0,r1,#1		Name Name Sptimer Sptimer1 Sptimer1	S Linked A	litera SoC FPGA Value	Size Acces	8		
<pre>struct of_mm_gpio_chip *mm_gc = to_of_mm</pre>	S:0x7F000034 E12FFF1E BX	tr goio set		sysmgr							
88 struct altera gpio_instance "chip = to_a	4 5:8x77888838 //RS	r3,APSR ; form	e	a la uarti							
89 90 spin lock irgsave(&chip->gpio lock, flag	\$:0x7F00003C CPSID 5:0x7F000040 CPSID	4		iti 🗁 usb0							
91	5:ex7F000044	r12,#1 r2,#0		🖲 😁 usbl							
92 /* Write to shadow register and output *	5:0x7F000048 E5902054 LDR	r2,[r0,#0x54]		😑 🈁 PIO_inst_0							
93 if (val) 94 chip->gpio state = 1 << gpio;	\$:0x7F00004C 1102111C ORRNE \$:0x7F000050 01C2111C BICEO	r1, r2, r12, LSL		PIO_inst_0_		0×00000000					
15 else	S:ex7F000054 E3902030 LDR	r1,r2,r12,LSL r2,[r0,#0x50]	r.	PIO_inst_0_		0×00000000					
<pre>96 chip->gpio_state &= ~(1 << gpio);</pre>	S:0x7F000058 E3801054 STR	r1,[r0,#8x54]		PIO_inst_0_I		8x83888888					
<pre>97raw_writel(chip->gpio_state, mm_gc->re 98</pre>	5:8x7F00005C E3021000 STR 5:8x7F000060 E121P005 MSR	r1,[r2,#0] CPSR c,r3		- e PIO inst 0		write only	32 WO				
<pre>99 spin_unlock_irgrestore(&chip->gpio_lock,</pre>	5:8x7F000064 8X	Lr		PIO_inst_0_		write only write only	32 WO				
100]			gpio_dir_in		B Co UART_inst_0						
102 /*		5:0x7F000068 13204004 PUSH	{r4}		B 🕞 JTAG_UART_in	st_0					
	1.2				B 😂 SYSID_inst_0						

ARM Compatibility a Given; FPGA Implementation a Difference

All the SoC FPGAs currently on the market leverage ARM processor IP, which generally includes support from the vast ecosystem for ARM processor software development tools. First and foremost, it is critical that the tools for these new devices be ARM-compatible and leverage the ARM ecosystem. However, each vendor deals differently with the added dimension of the FPGA portion of the device. This particularly impacts the following:

Whole-Chip Debug

With SoC FPGAs, the SoC is no longer pre-defined, and the debugging tools must support a number of new constructs, namely:

- · Adapt to changing user-defined peripherals implemented in the FPGA
- · Test software functions that include hardware acceleration blocks implemented in the FPGA
- Debug custom logic blocks in the FPGA that implement proprietary algorithms

Traditional software debugging tools were not designed for fluctuating hardware functions, and traditional FPGA tools have no 'hooks' back to the software tools. To bridge this debugging chasm, a toolset must provide:

- · Whole-chip visibility of both the processor and the FPGA subsystems
- · Cross-triggering and in-system trace between CPU and FPGA subsystem
- · System-wide monitoring for software, CPU hardware, and FPGA hardware events
- Performance profiling

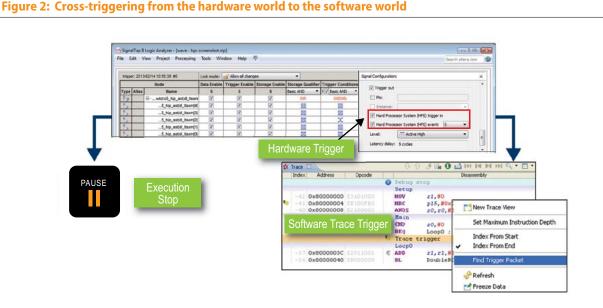
Ideally, the debugging tool should be as flexible as the FPGA. This vision is called 'FPGA-adaptive debugging', and it has become a reality with the Altera SoC FPGA development flow.

FPGA-adaptive debugging means that software debug tools automatically adapt to changes in the hardware due to changes in the FPGA logic. As the hardware engineer iterates through various FPGA configurations, the software debug view updates automatically, with all FPGA-based peripherals automatically appearing in the register view.

CoreSight Compliant Cross-Triggering

Finding the cause of a bug is much easier if the processor subsystem and FPGA subsystem can cross-trigger from code to waveform, or from waveform to code, enabling the development team to find and track how and why a particular condition occurred in the system. *Figure 2* shows a cross-triggering example from the ARM Development Studio-5 (DS-5^m) Altera Edition Toolkit software. When coupled with waveform views from the Altera SignalTap^m II Logic Analyzer, cross-triggering, trace, and global time-stamping are valuable features for IP verification, custom driver development, and the system integration portion of a project.

Though various debug tools and associated cables are available for the ARM processor and FPGA fabric, it can be difficult to manage two separate debugging frameworks at a system level. Correlating events becomes next to impossible. Most developers prefer a single (and low-cost) JTAG cable that supports both hardware targets (the FPGA and the CPU subsystem). Using one cable and a commonlyunderstood software interface provides an efficient, easy-to-use means for driver developers, board engineers and FPGA designers to work together to bring up the entire system.



Besides finding the location of fault, it is also valuable to find out exactly how and why the system entered the faulty state. The ARM CoreSight[™] System Trace Module (STM) contained in Altera SoC FPGAs enables tracking of CPU-based software events as well as user-defined system level states such as 'low power mode' or 'high performance mode'. The application software can issue hardware and software event "bread crumbs" as the system executes over time to monitor system behavior. In an FPGA-adaptive debugging environment, the STM enables event monitoring in both the CPU and FPGA domains without stopping the system or affecting execution performance.

Multicore Debug

As the embedded world moves to multicore, development tools must follow suit. Developing software for multicore platforms is more complicated than single core. Choosing on which core to set a breakpoint; determining on which core the software is running at any particular time becomes critical for multicore debug.

It is essential to be able to control and monitor the cores simultaneously as well as independently. In some cases it may be necessary to stop both cores on a breakpoint; but in others, it may be preferable to let one core keep running on a breakpoint, too. It is also valuable to have visibility to the software running on each core. Ideally a debugger and analysis tools, both built from the ground-up for multicore systems, should be used.

GNU tools were designed in the single-core era; the GNU Debugger (GDB) works well, but only on a single core at a time. When using a GDB-based debugger on a multicore system, breakpoints can be set up across multiple cores. When software hits a breakpoint, only the core where the breakpoint occurred can be viewed during the debug session, which is extremely limiting for multicore debugging.

Table 1: In-System Debugging and Development Tool Features for SoC FPGA Devices								
Function/Feature	Altera SoC EDS (with ARM DS-5 Altera Edition)	Vendor B's Debug Tools						
Versions Compared	13.1	2013.3						
FPGA-Adaptive Debugging	Yes	No						
All ARM Processor and FPGA Tools Operate Over Single USB Cable	Yes	No						
Auto Display of Peripheral Registers	Yes	No						
Display of VFP and Neon Registers	Yes	No						
Debug: Single-Step, Watchpoints, etc.	Yes	Yes						
CPU⇔FPGA CoreSight Compliant Cross-Triggering	Yes	No Vendor proprietary						
CPU⇔FPGA Cross-Triggering with Timestamps and Trace Data Stream	Yes ARM CoreSight™ compliant using System Trace Macrocell (STM)	No Available with purchase of additional third-party hardware and software						
Processor Trace Support	Yes	No Requires additional 3rd-party hardware and software						
Trace Buffer	32 KB	4 KB						
Route Trace Packets to Alternative Destinations (e.g. DRAM or high-speed transceiver)	Yes Coresight Embedded Trace Router	No						
Route Trace Packets to External Trace Probe	Yes	Yes						
FPGA Information Included in ARM Trace Stream	Yes Uses ARM CoreSight System Trace Macrocell	Yes Vendor proprietary solution						
Native Linux Support for Hardware-Assisted Trace	Yes Kernel and Application	No						
Concurrent Multicore Debugger	Yes ARM DS-5 Specifically designed for multicore systems	No						
Multicore Debugging in Asymmetric Multiprocessing (AMP) Applications	Yes	Yes						
Multicore Debugging with Symmetric Multiprocessing (SMP) Operating Systems	Yes	No						
Linux Kernel Awareness	Yes	No						
Code Profiling	Yes ARM Streamline including processor, FPGA, and power profiling. No instrumentation required	No						
Semi-Hosting Support (communication between host and ARM processors over JTAG)	Yes	No						
FPGA Logic Analyzer	SignalTap II Logic Analyzer	Yes						
Bare-Metal Application Development	Modifiable hardware libraries with friendly, open BSD licensing	Vendor proprietary BSP project build						
Hardware VFP and NEON Compiler Support	Yes (Linux) Support for Bare Metal compiler available in version 14.0	Yes (Linux/Bare Metal)						

Conclusion

To provide hardware and software teams the tools they need, and to keep projects on schedule and on budget, the right in-system debugging tool is critical. This Architecture Brief described how the Altera solution based on the ARM DS-5 Altera Edition Toolkit offers simultaneous insight and control into both the ARM processor and FPGA logic portions of the SoC FPGA, enabling project teams to implement advanced features in these new devices while keeping schedules on track.



Copyright © 2014 Altera Corporation. All rights reserved. Altera, the stylized Altera logo, speciác device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. July 2014 AB20-01226