SW Development for Altera SoC Devices Workshop

Altera SW SoC Workshop Series

- SW Workshop #1 – Altera SoC SW Development Overview
- SW Workshop #2 – Introduction to Linux on Altera SoC
- SW Workshop #3 – Developing Drivers for Altera SoC Linux

Welcome. Here's What You Can Expect Today

- An SoC SW development *quick-start* to quickly evaluate the SoC tool flow
  - SoC EDS tool suite and debug capability
  - Tool chain for bare metal development
  - Take-home lab to test drive the development tools
- An overview of the SoC HW for SW developers
- An overview the SoC SW ecosystem
- A useful lead-in to the detailed SoC SW training

Agenda

- Informational Resources
- SoC Device Overview
- Boot Process
- Hardware Development Flow and Tools
- Software Development Flow and Tools
  - DS-5 Altera Edition
- SoC Physical Address Maps
- Bare Metal Software Overview
- Linux Software Overview
- Supported OSes
- Preloader Generator and u-boot
- SD Card Manipulation
- System Console
- LAB overview

Information Resources

Several Ways to Learn!

- **Instructor-led training**
  - Face to face with an Altera expert Training Engineer
  - 20+ courses to choose from (8 hour classes)

- **Virtual classes (taught via WebEX)**
  - Can ask questions to Altera expert Training Engineer
  - Course content same as instructor-led classes (1/2 day sessions)

- **Online training (free and always available)**
  - 200+ topics available (~30 minutes in length)

- **Videos (free and always available)**
  - YouTube videos (~4 minutes each)

SoC Classes Available

- Instructor-led or virtual classes
  - Designing with an ARM-based SoC
  - Developing Software for an ARM-based SoC
- Online classes
  - Hardware Design Flow for an ARM-based SoC
  - Software Design Flow for an ARM-based SoC
  - SoC Hardware Overview: the Microprocessor Unit
  - SoC Hardware Overview: Interconnect and Memory
  - SoC Hardware Overview: System Management, Debug, and General Purpose Peripherals
  - SoC Hardware Overview: Flash Controllers and Interface Protocols
  - SoC Bare-metal Programming and Hardware Libraries
  - Getting Started with Linux for Altera SoCs

Essential SoC Hardware Documentation Resources

- Hard Processor System Technical Reference Manuals
  - Available in Device Handbooks:
    - https://www.altera.com/products/soc/portfolio/cyclone-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-10-soc/support.html
  - Contain Functional Descriptions Peripheral
  - Contain Control Register Address Map and Definitions
    - These are also available online at the links above in HTML and PDF formats
- HPS SoC Boot Guide
  - Cyclone V SoC & Arria V SoC: AN709 - HPS SoC Boot Guide
  - Arria 10 SoC: included in HPS TRM in Arria 10 Device Handbook
    - Arria 10 SoC secure booting: AN759 – Arria 10 SoC Secure Boot User Guide
- ARM Documentation Site
  - Documentation available for all ARM IP
    - Cortex-A9 & A53 MP Cores, FPU, NEON, GIC, ARM Peripherals, etc.
  - Requires free registration
  - Refer to HPS TRM for IP core names and revision information
  - http://infocenter.arm.com/help/index.jsp

Essential SoC Software Documentation Resources

- Altera SoC Embedded Design Software (SoC EDS) Tools
  - User Guide:
    https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_soc_eds.pdf
    - Linux & Baremetal Software Development Tools Overview
    - HPS Preloader User Guide
    - HPS Flash Programmer User Guide
    - SD Card Boot Utility
  - Getting Started Guides: Preloader, Linux, Bare Metal, Debug, HW Library
    http://www.alterawiki.com/wiki/SoCEDSGettingStarted
  - SoC HPS Release Notes
  - SoC Abstraction Layer (SoCAL) API Reference
    - <SoC EDS install dir>/ip/altera/hps/altera_hps/doc/socal/html/index.html
  - Hardware Manager API Reference
    - <SoC EDS install dir>/ip/altera/hps/altera_hps/doc/hwmgr/html/index.html
  - GCC Documentation
    - <SoC EDS install dir>/ds-5/documents/gcc/getting_started.html
  - Bare Metal Compiler
    - <SoC EDS installation directory>/host_tools/mentor/gnu/arm/baremetal/share/doc/sourceryg++-arm-altera-eabi

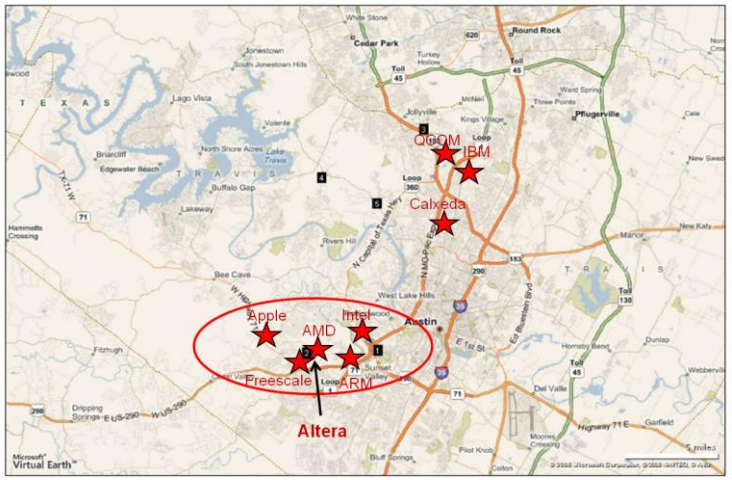
Essential SoC Software Tools Online Videos

- ARM DS-5 Altera Edition Toolchain
  - https://youtu.be/HV6NHr6gLx0
- DS-5 Altera Edition: Bare-metal Debug and Trace
  - https://youtu.be/u_xKybPhcHI
- DS-5 Altera Edition: FPGA-adaptive Linux Kernel Debug and Trace
  - https://youtu.be/lrR-SfVZd18
- Debugging Linux applications on the Altera SoC with ARM DS-5
  - https://youtu.be/ZcGQEjkYWOc
- FPGA-adaptive debug on the Altera SoC using ARM DS-5
  - https://youtu.be/2NBcUv2TxbI
- Streamline Profiling on Altera SoC FPGA. Part 1 - Setup
  - https://youtu.be/X-k9lmXQTio
- Streamline Profiling on Altera SoC FPGA. Part 2 - Running Streamline
  - https://youtu.be/Tzbd7qldKqY

SoC Device Overview

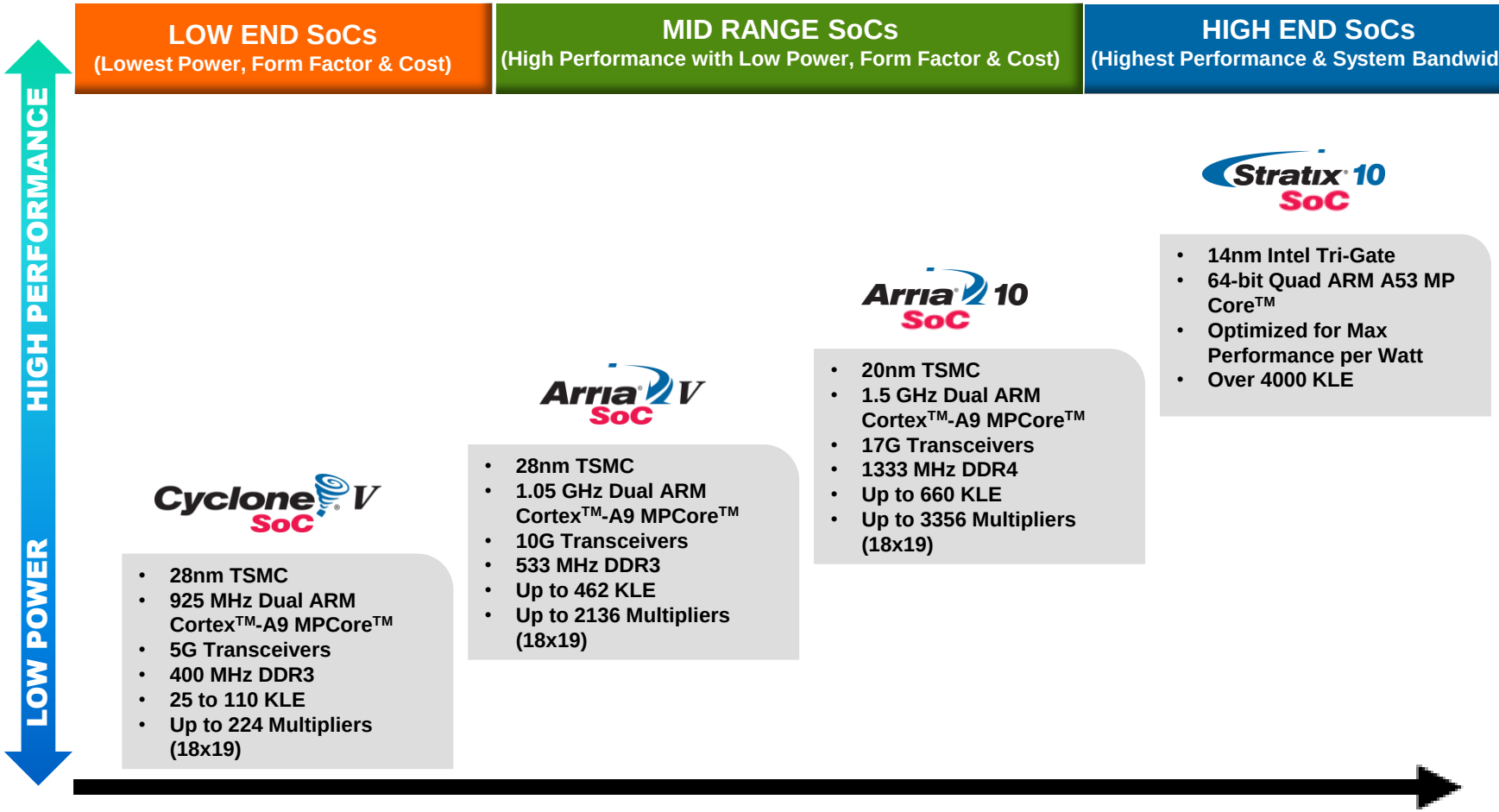
Altera Investment in Embedded Technologies

- Altera established Austin Technology Center (ATC) in 2011
- Altera's primary embedded engineering center
- Austin provides access to one of the richest embedded processing talent bases in the world



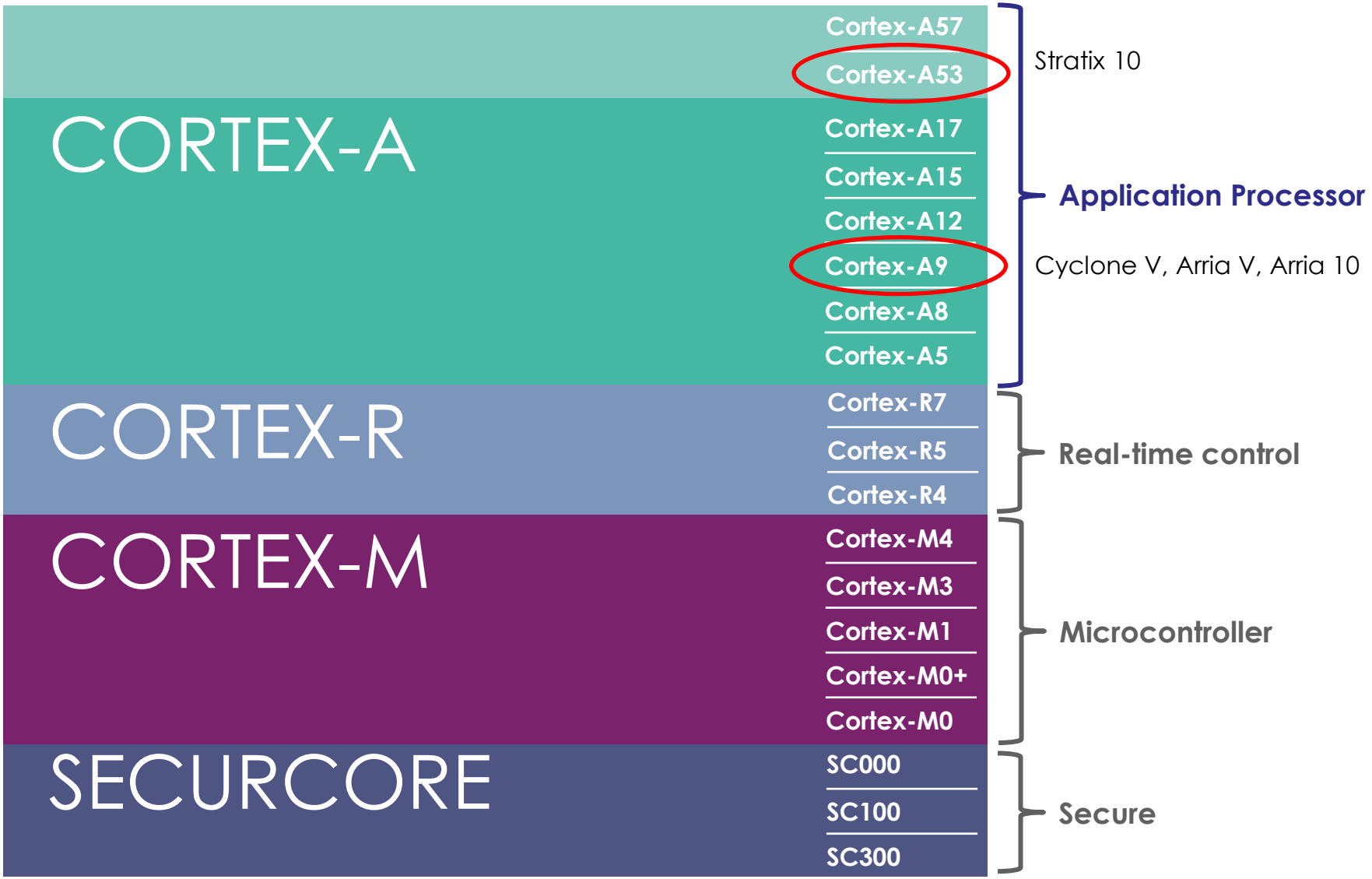
ALTERA
now part of Intel

Altera SoC Product Portfolio



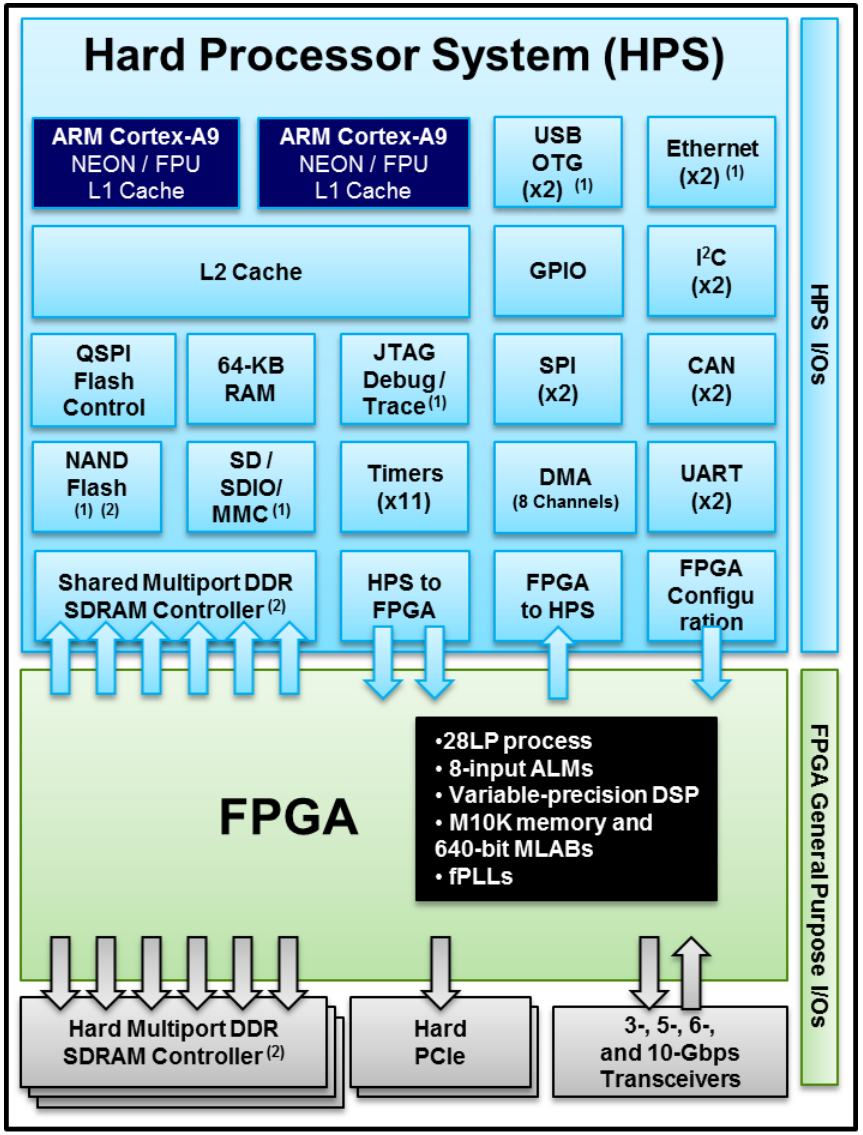
SoC devices available across entire product portfolio …

ARM Public Processor Offering



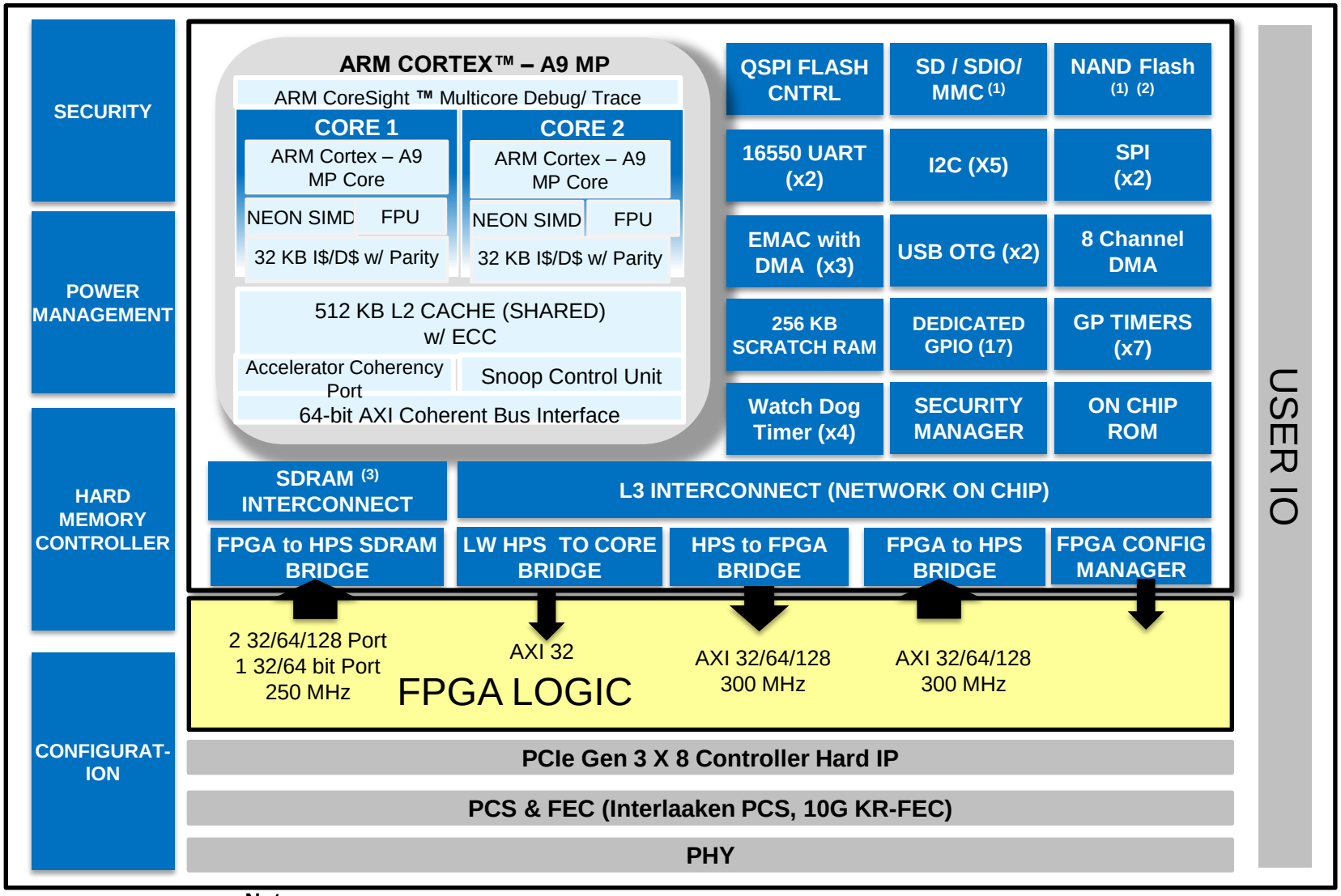
28nm SoC System Architecture

- Processor
  - Dual-core ARM® Cortex™-A9 MPCore™ processor
  - Up to 5,250 MIPS (1050 MHz per core maximum)
  - NEON coprocessor with double-precision FPU
  - 32-KB/32-KB L1 caches per core
  - 512-KB shared L2 cache
- Multiport SDRAM controller
  - DDR3, DDR3L, DDR2, LPDDR2
  - Integrated ECC support
- High-bandwidth on-chip interfaces
  - > 125-Gbps HPS-to-FPGA interface
  - > 125-Gbps FPGA-to-SDRAM interface
- Cost- and power-optimized FPGA fabric
  - Lowest power transceivers
  - Up to 1,600 GMACS, 300 GFLOPS
  - Up to 25Mb on-chip RAM
  - More hard intellectual property (IP): PCIe® and memory controllers



Notes:
(1) Integrated direct memory access (DMA)
(2) Integrated ECC

Arria 10 HPS Block Diagram



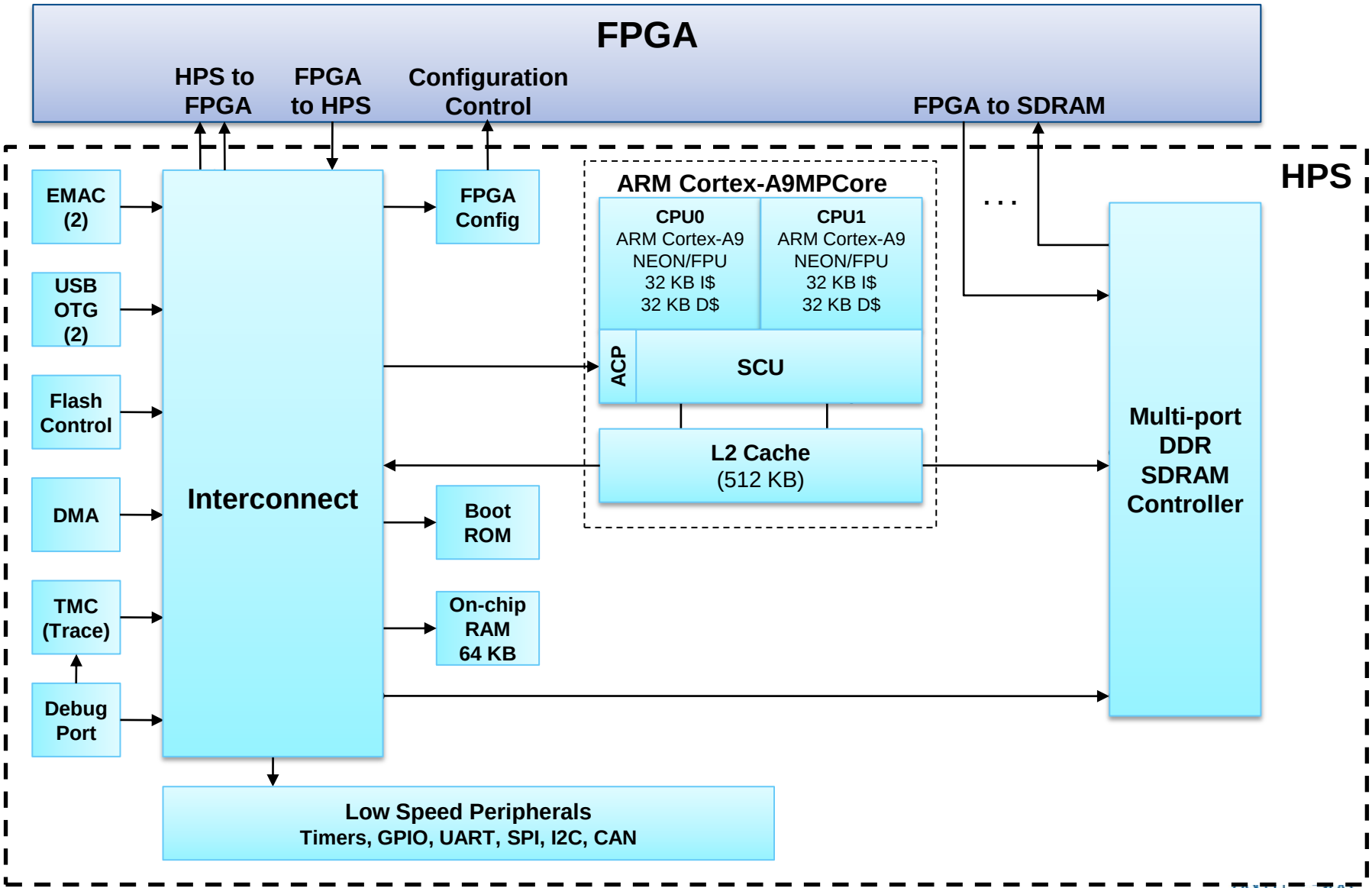
Notes:

(1) Integrated direct memory access (DMA)

(2) Integrated ECC

(3) DDR3/4 SDRAM Support for HPS Memory

High-Level Block Diagram



A Comparison: Cyclone V SoC, Arria V SoC, Arria 10 SoC

Metric	Cyclone V SoC	Arria V SoC	Arria 10 SoC
Technology	28nm	28nm	20nm
Processor Performance	925 MHz	1.05 GHz	1.5 GHz
Total Power Dissipation	100%	100%	60% (40% Lower)
Max PCI Express Hard IP	Gen 2 x4	Gen 2 x8	Gen 3 x8
Memory Devices Supported	DDR2, DDR3, DDR3L, LPDDR2	DDR2, DDR3, DDR3L, LPDDR2	DDR4/3, LPDDR2/3, QDRIV, RLDRAM III, Hybrid Memory Cube*
Max. HPS DDR Data-Width	40-bit (32-bit + ECC)	40-bit (32-bit + ECC)	72-bit (64-bit + ECC)
EMAC Cores	EMAC x 2	EMAC x 2	EMAC x 3
NAND Device Supported	8-bit	8-bit	8-bit and 16-bit
SD/MMC devices supported	SD/SDIO/MMC	SD/SDIO/MMC	SD/SDIO/MMC 4.5 with eMMC
FPGA Logic Density Range (LEs)	25 - 110K	370 - 450K	160 - 660K
FPGA Core Performance	260 MHz	307 MHz	500 MHz

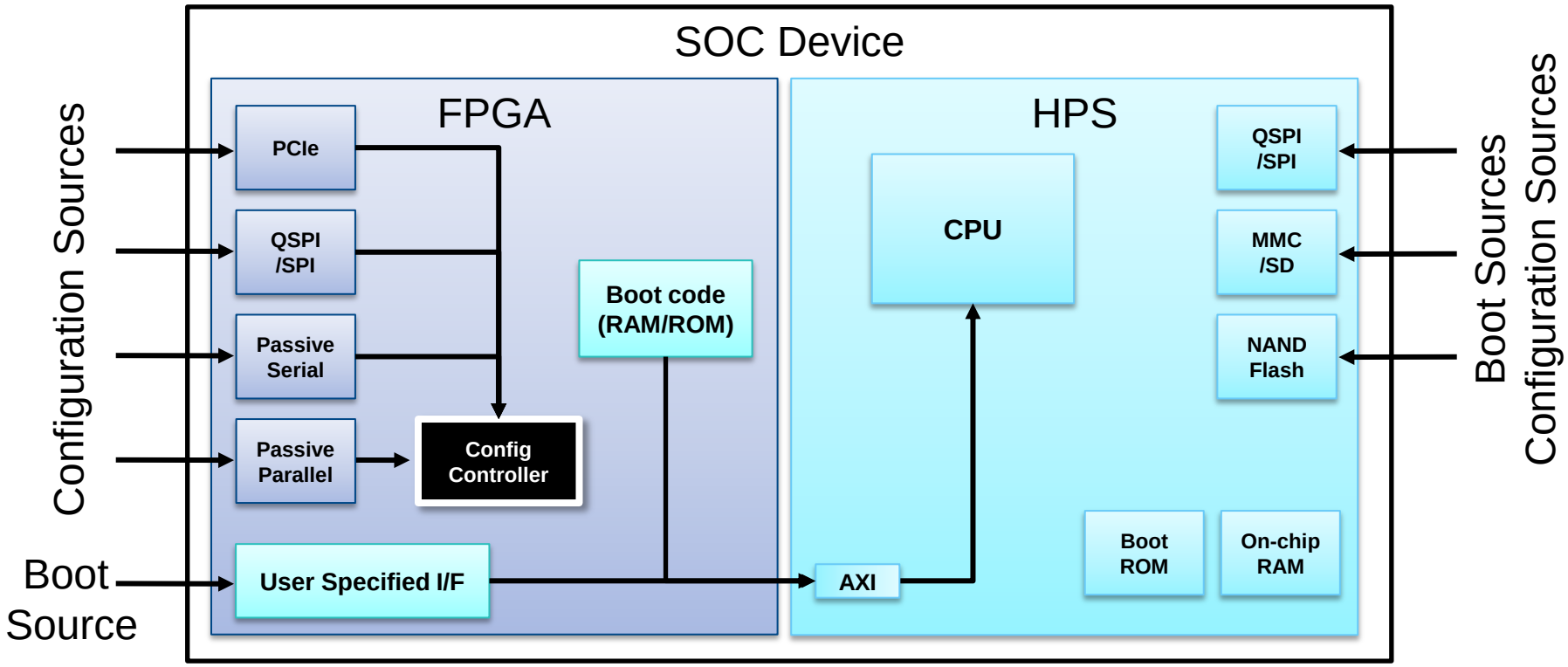
*** Listed memory protocols are supported by FPGA EMIF interface, the HPS EMIF interface supports a subset of these.**

Learn more about the HPS hardware

- Overview
  - http://www.altera.com/devices/processor/soc-fpga/overview/proc-soc-fpga.html
- SoC handbooks
  - Choose the device family you are interested in from the links below and then locate the “Hard Processor System Technical Reference Manual” for that device.
    - https://www.altera.com/products/soc/portfolio/cyclone-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-10-soc/support.html

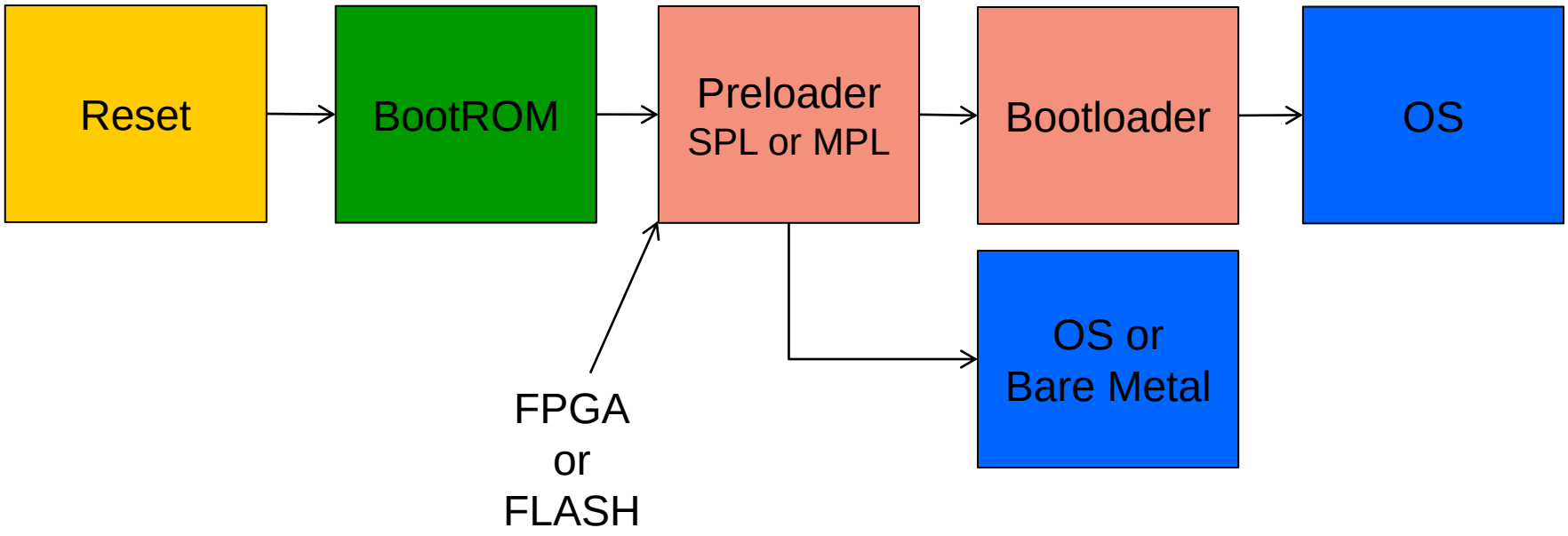
Boot Process

Altera SoC FPGA Boot & Configuration Options



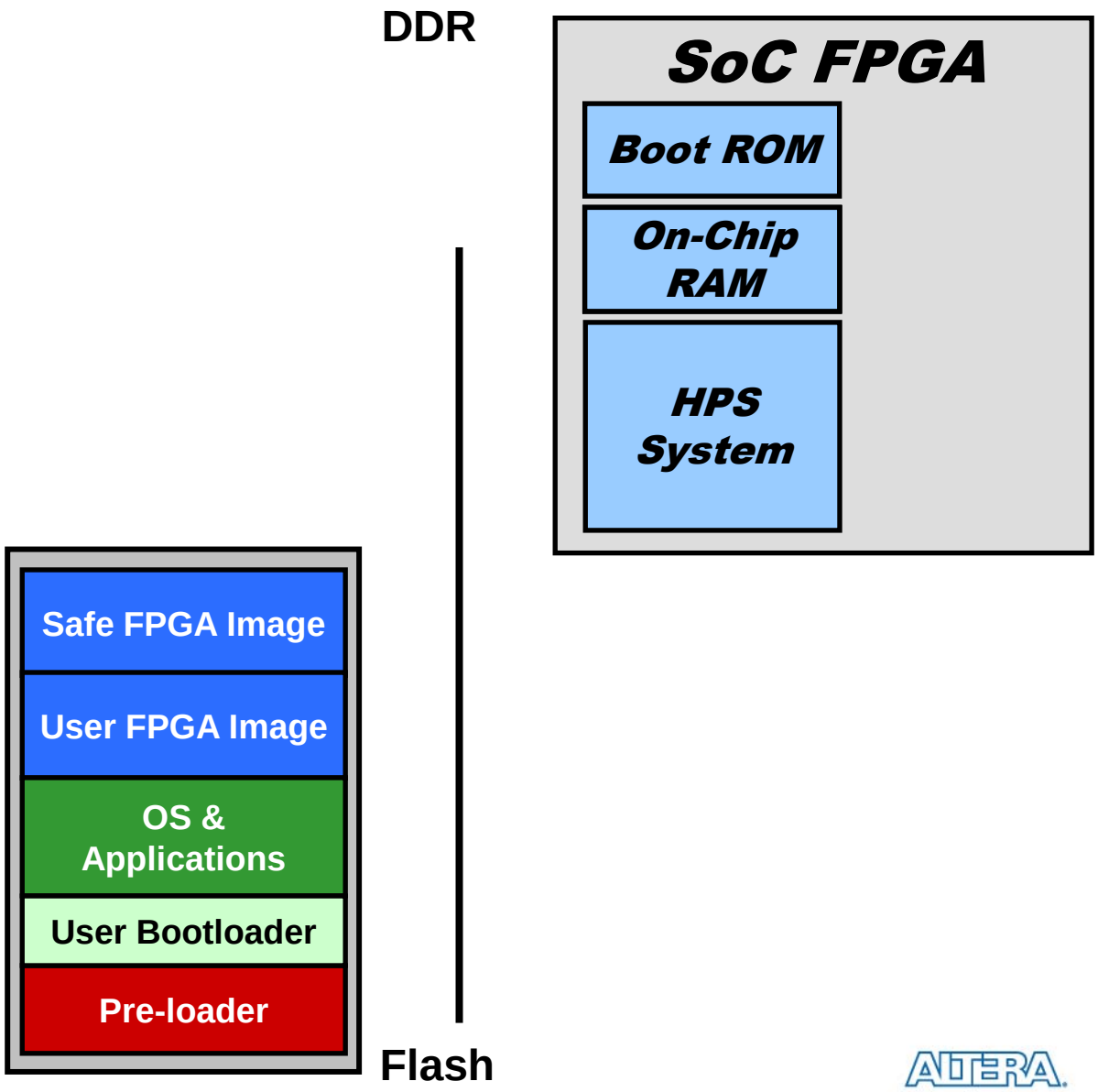
Boot Stages – Cyclone V & Arria V

- General Boot Process



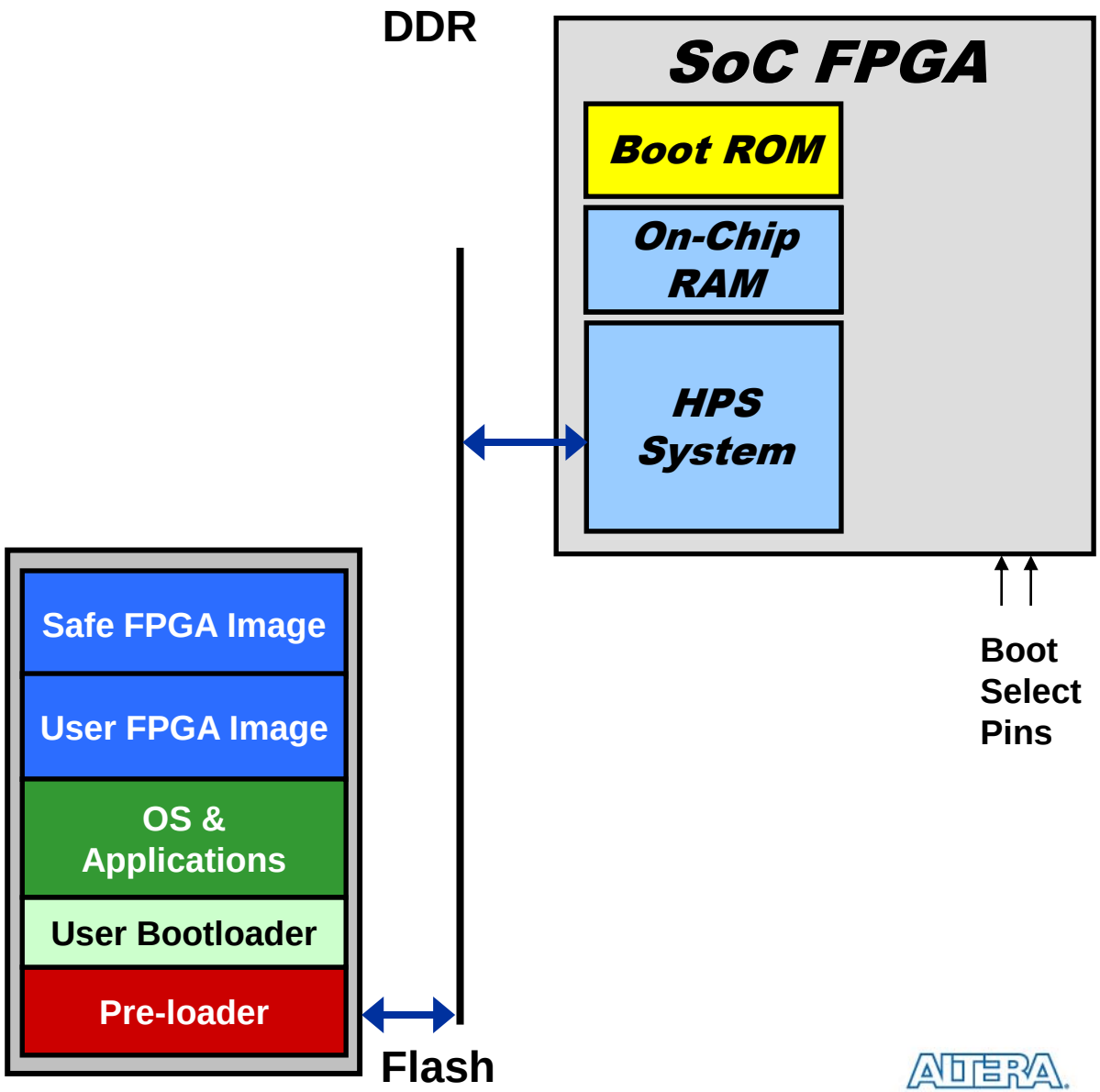
Boot Stages – Cyclone V & Arria V

1. Reset



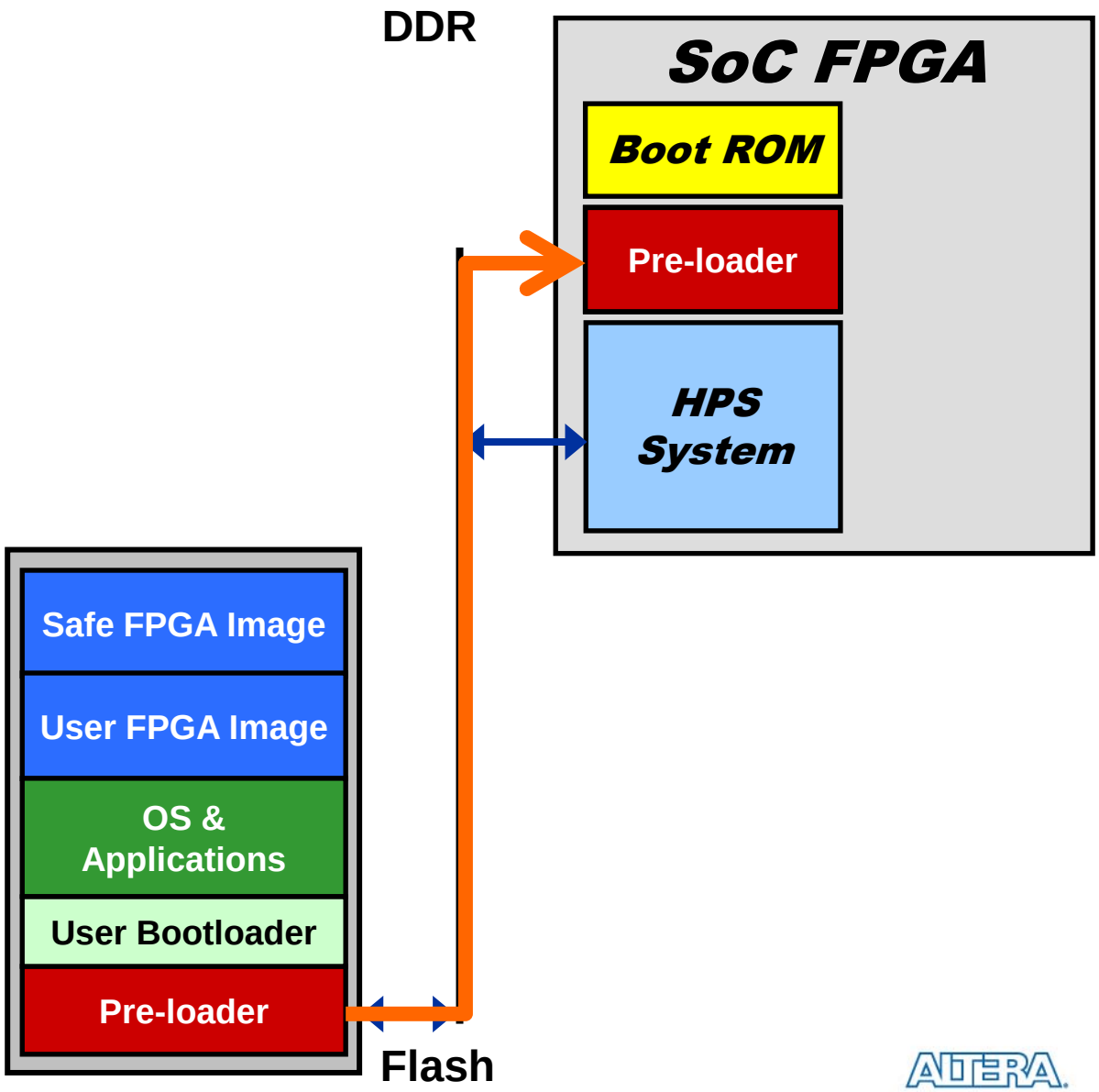
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins



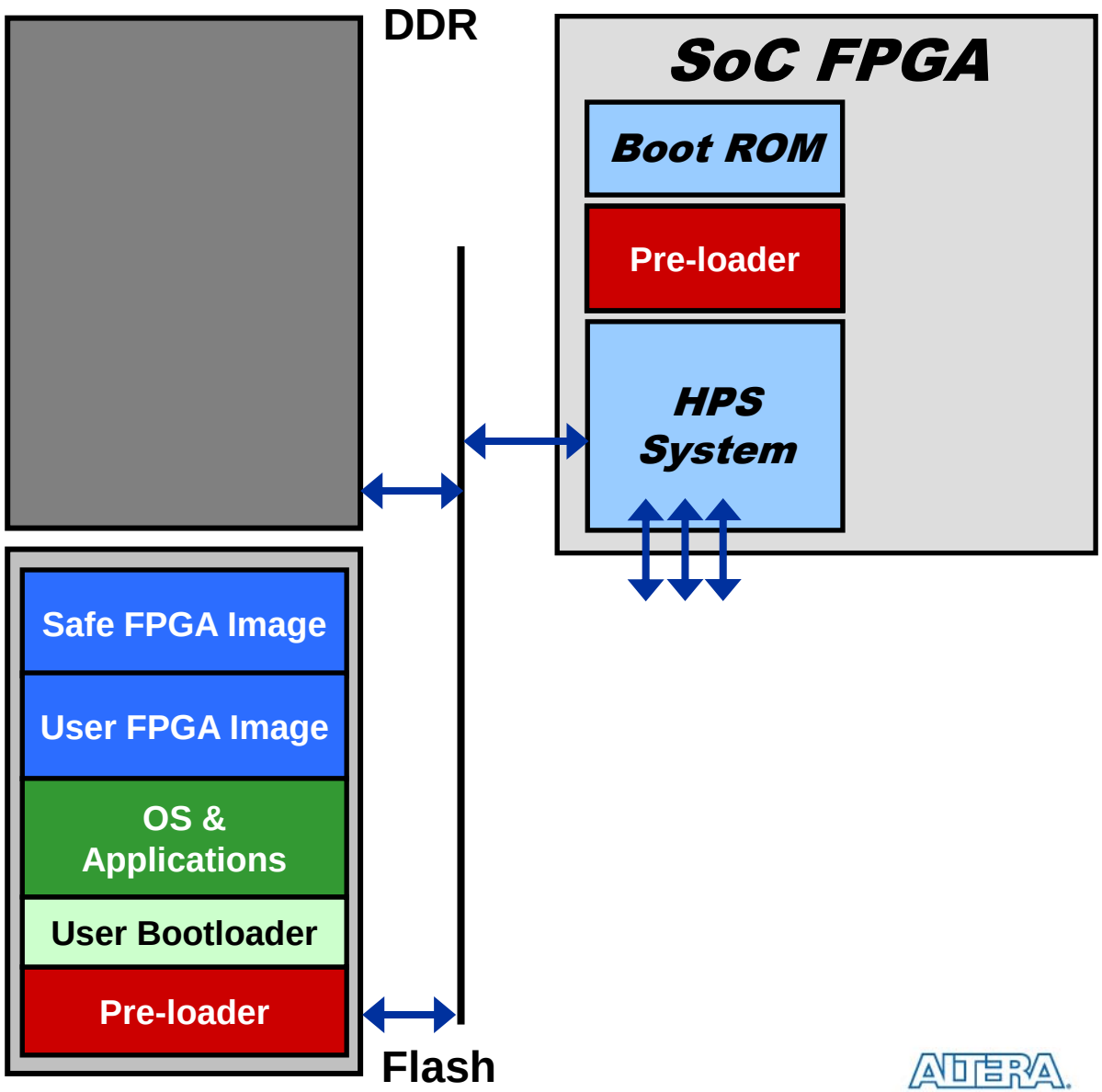
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM



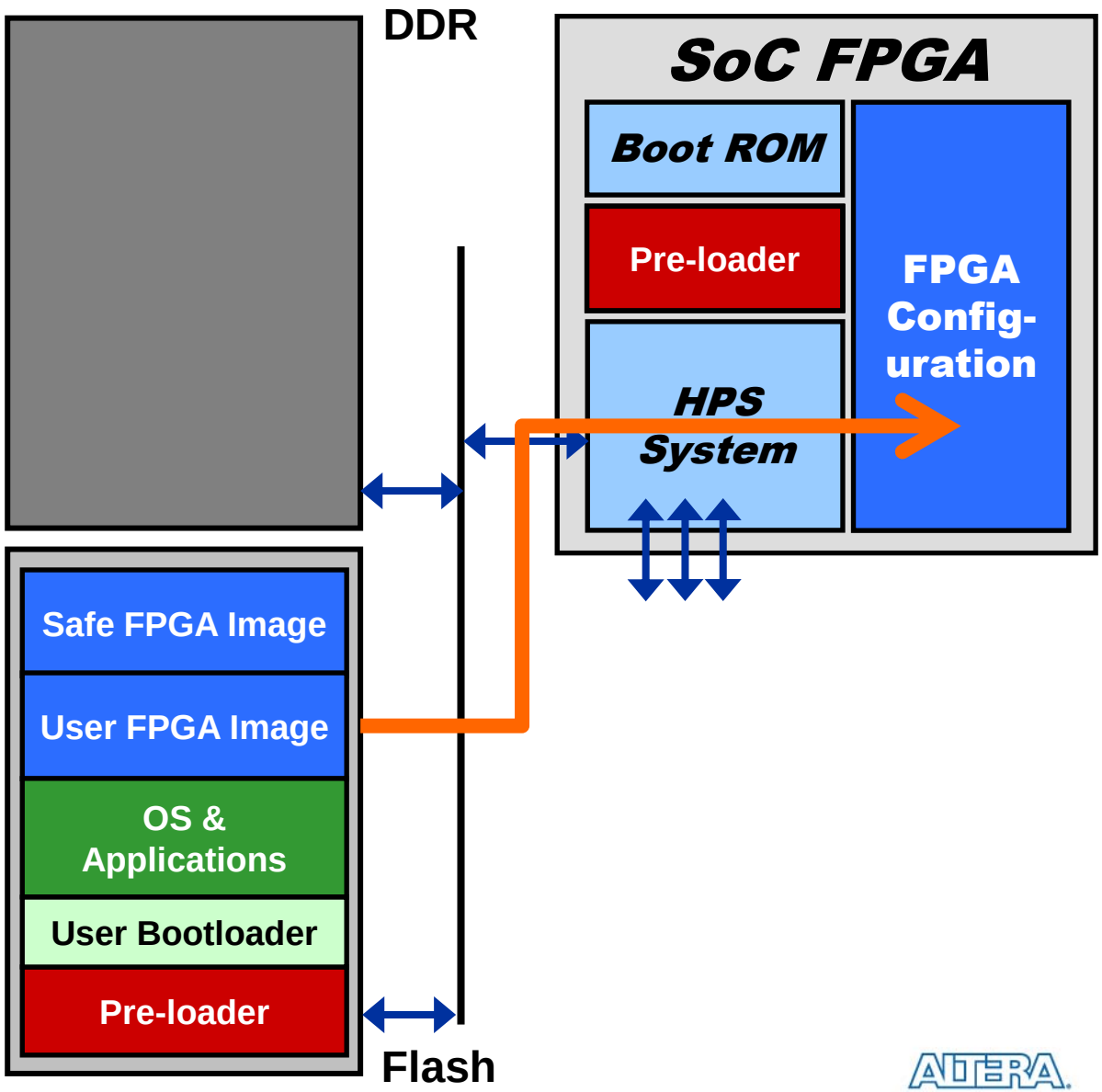
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR



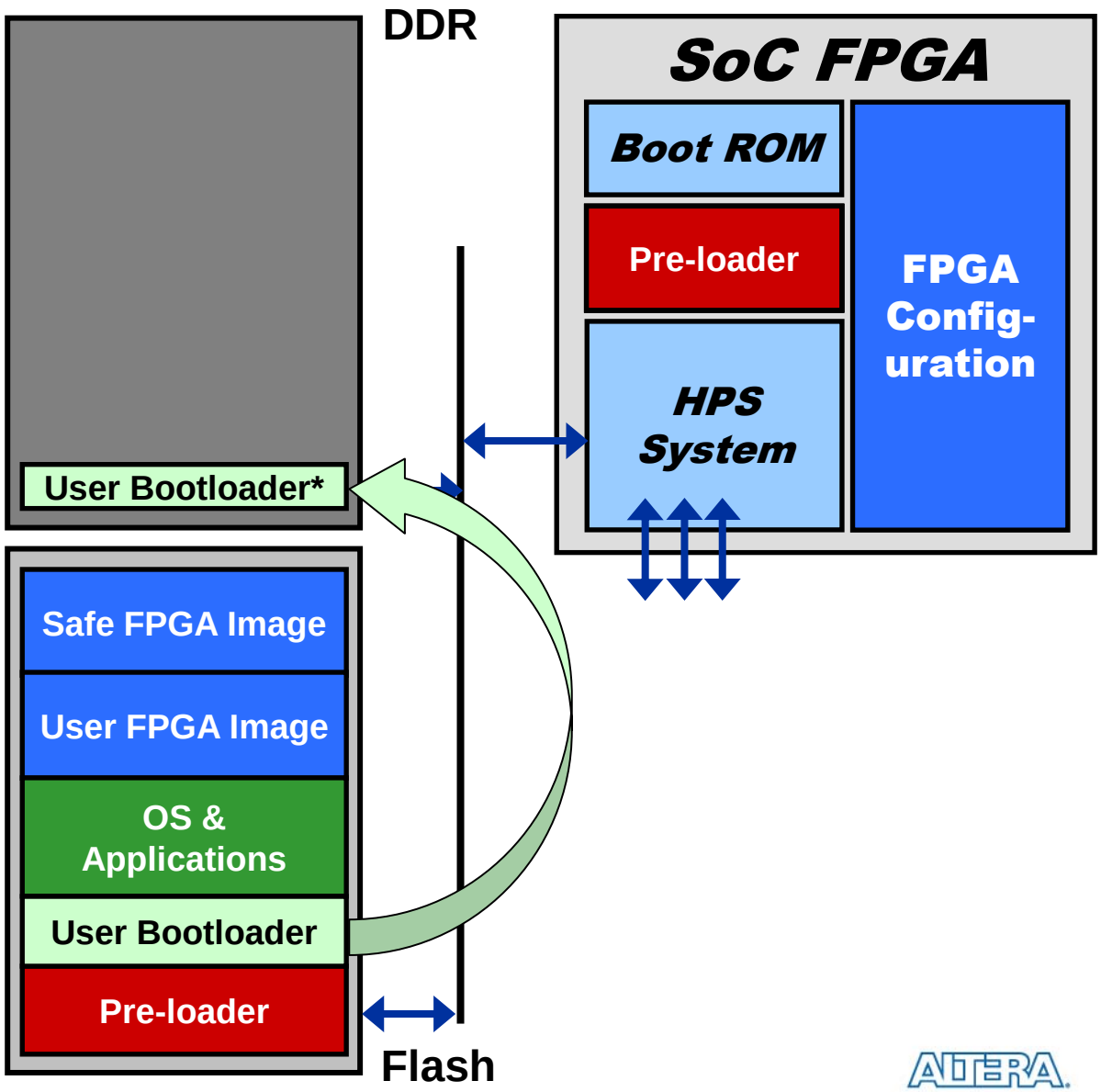
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)



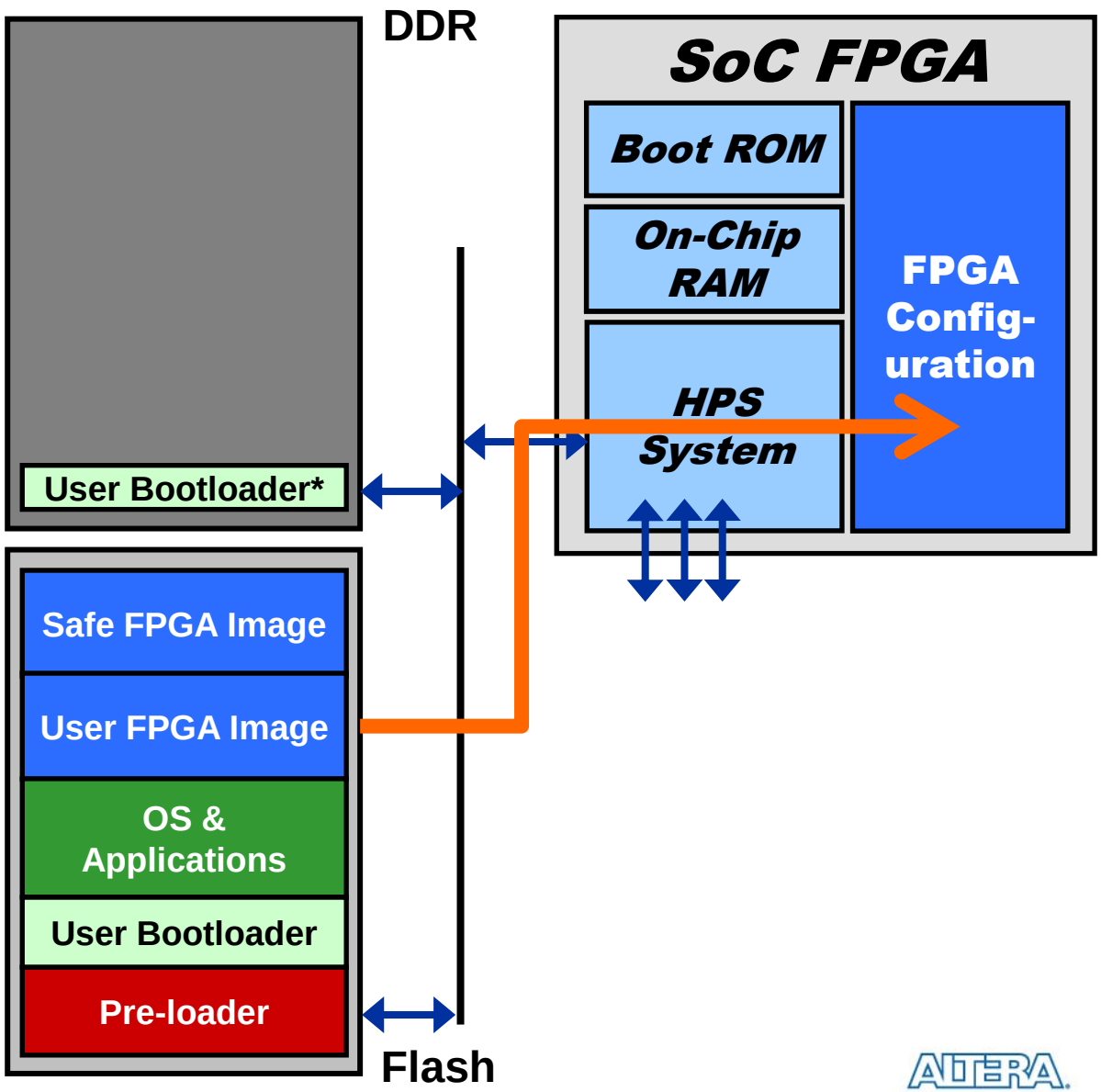
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy User Bootloader into DDR RAM



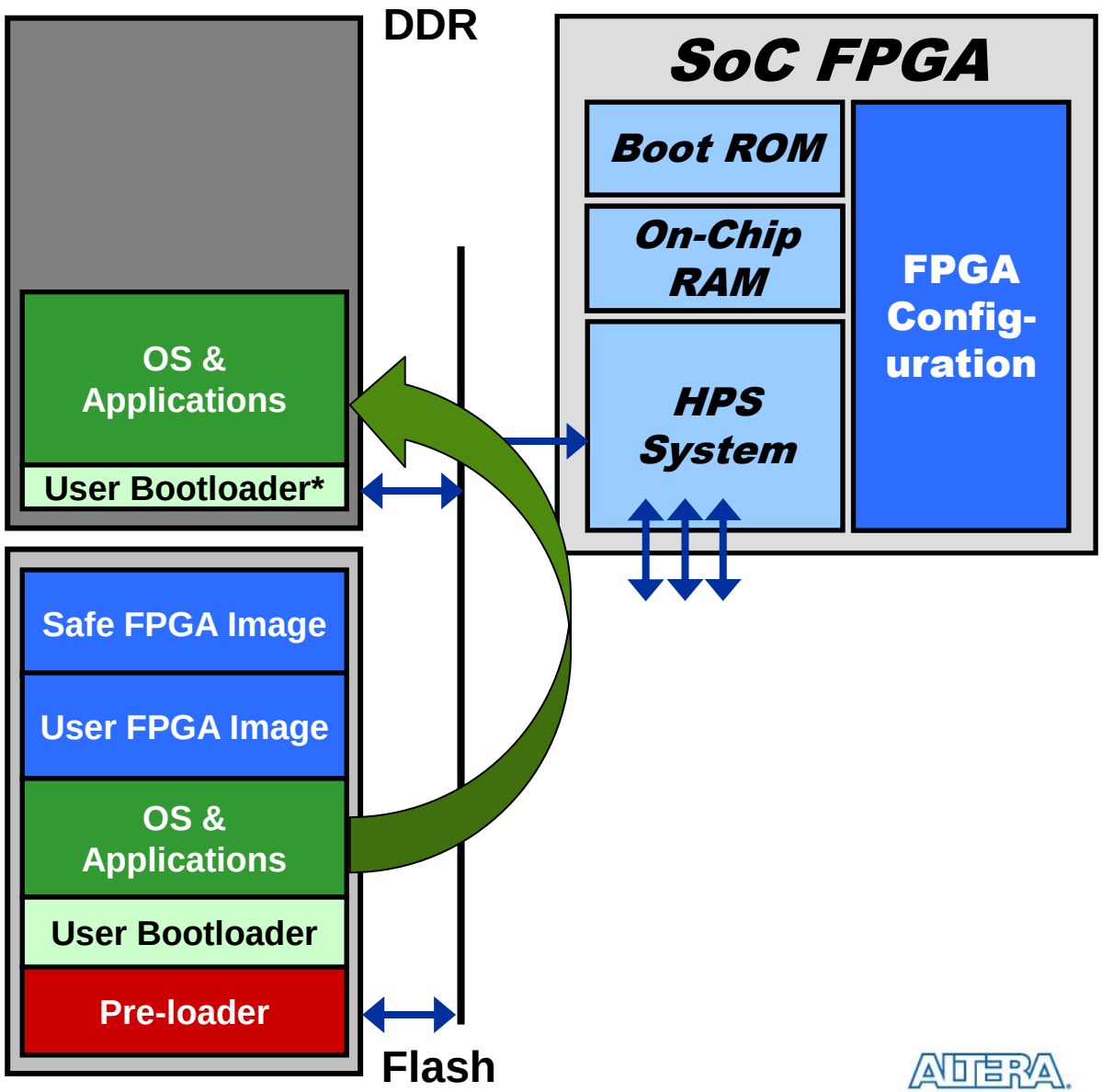
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)



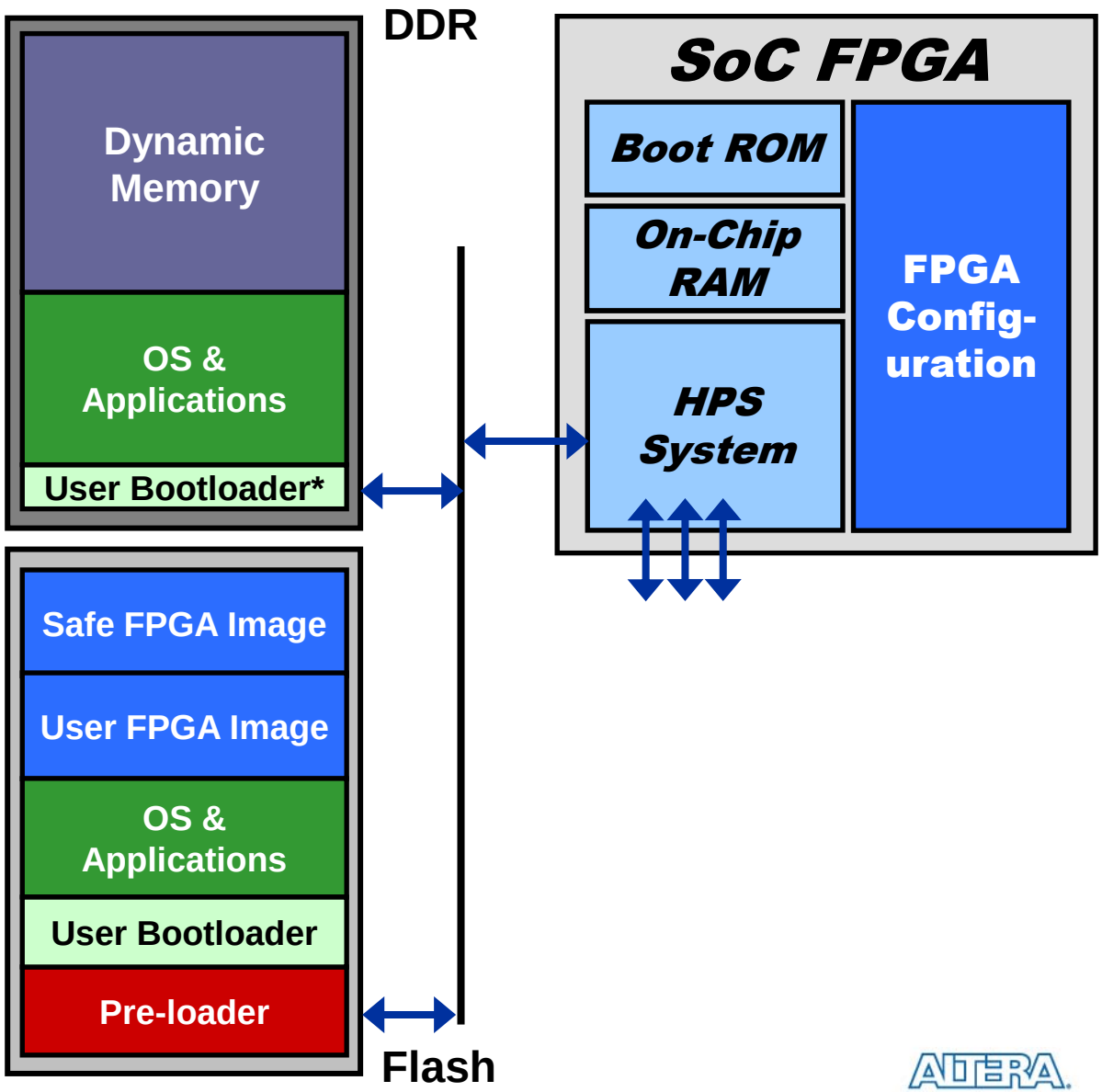
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM



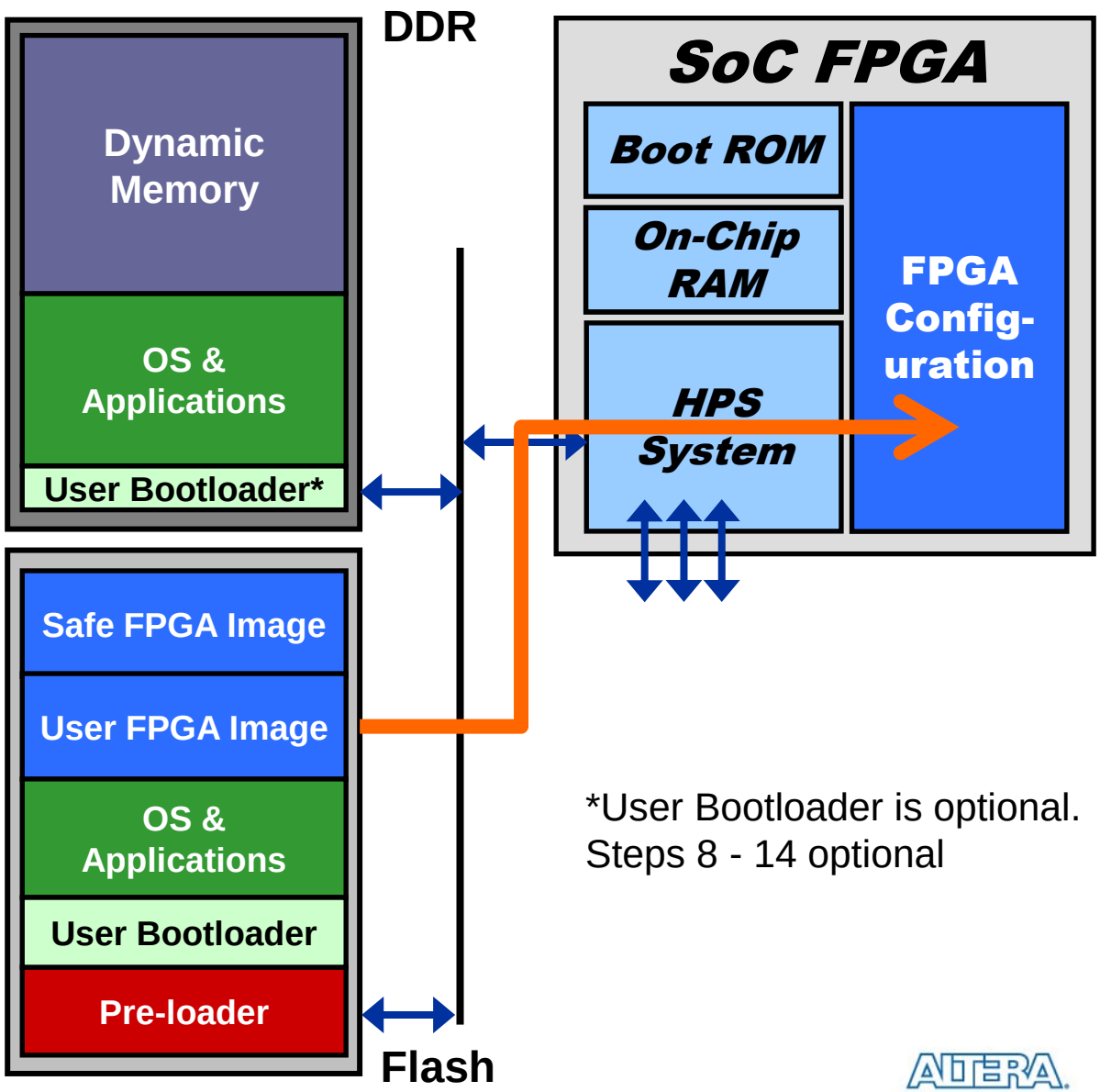
Boot Stages – Cyclone V & Arria V

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM
12. Run OS
13. Run applications



Boot Stages – Cyclone V & Arria V

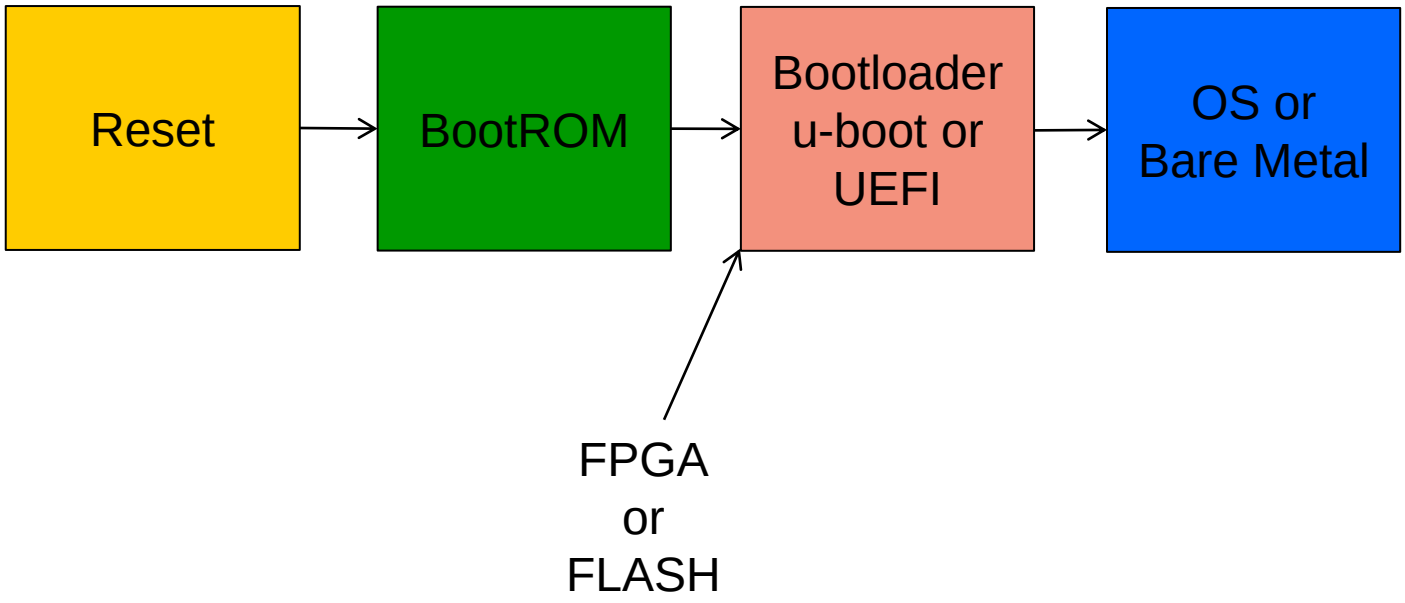
1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run Pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM
12. Run OS
13. Run applications
14. Configure FPGA (optional)



*User Bootloader is optional.
Steps 8 - 14 optional

Non-Secure Boot Stages – Arria 10

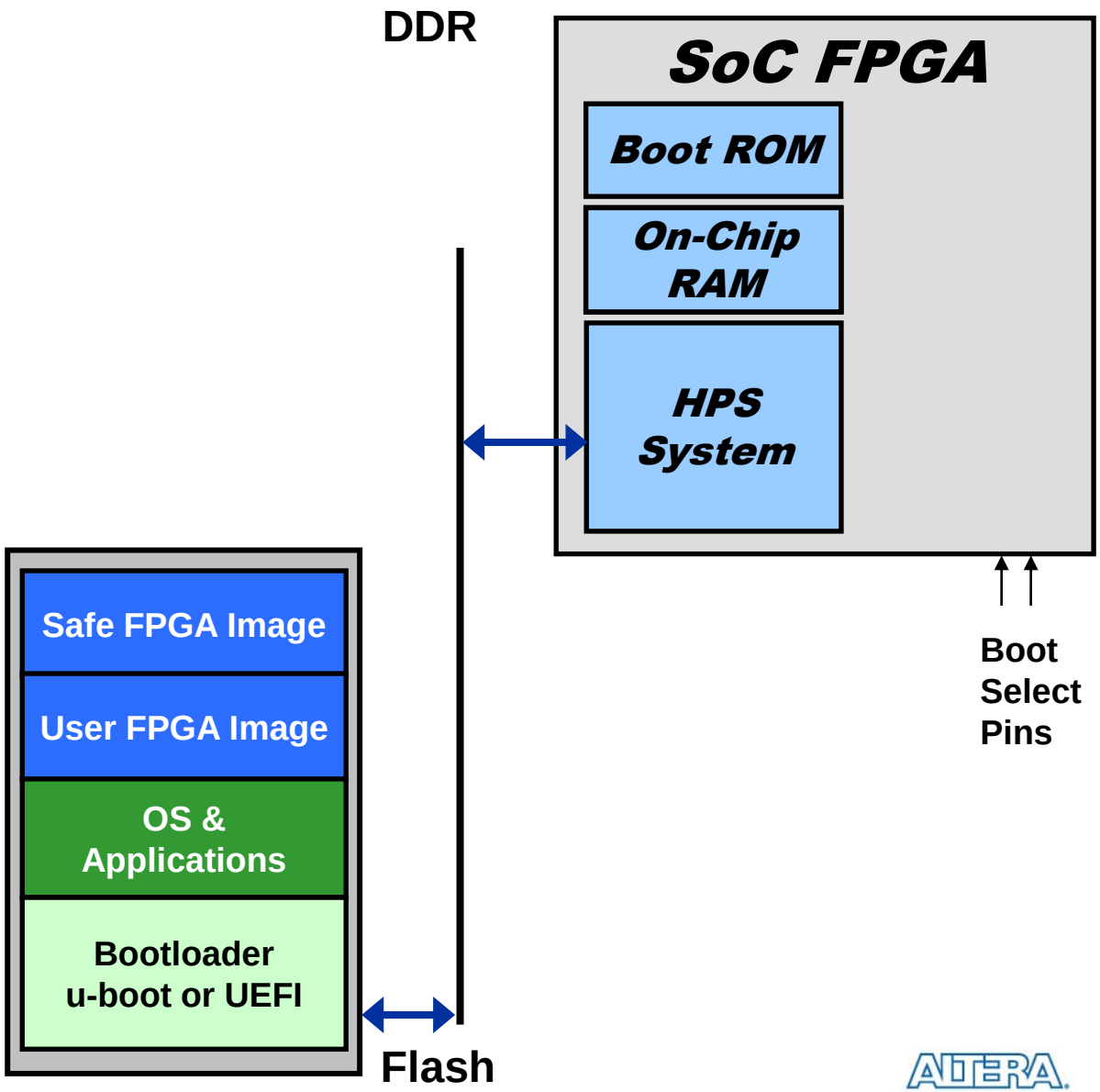
- General Boot Process for Non-Secure Boot Flow
  - Arria 10 supports secure and non-secure boot flows



Enabled by larger on-chip RAM in Arria 10 HPS

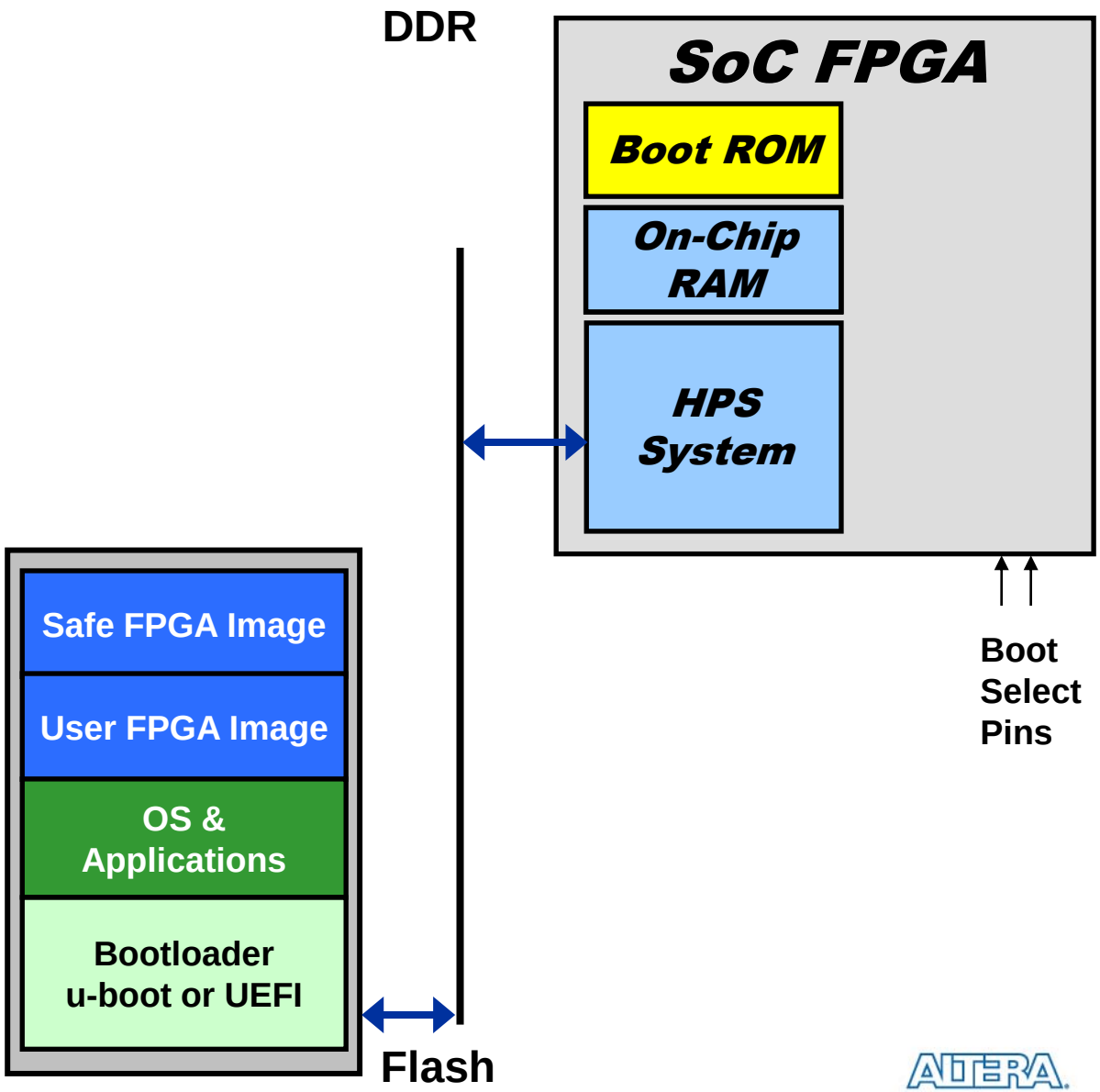
Non-Secure Boot Stages – Arria 10

1. Reset



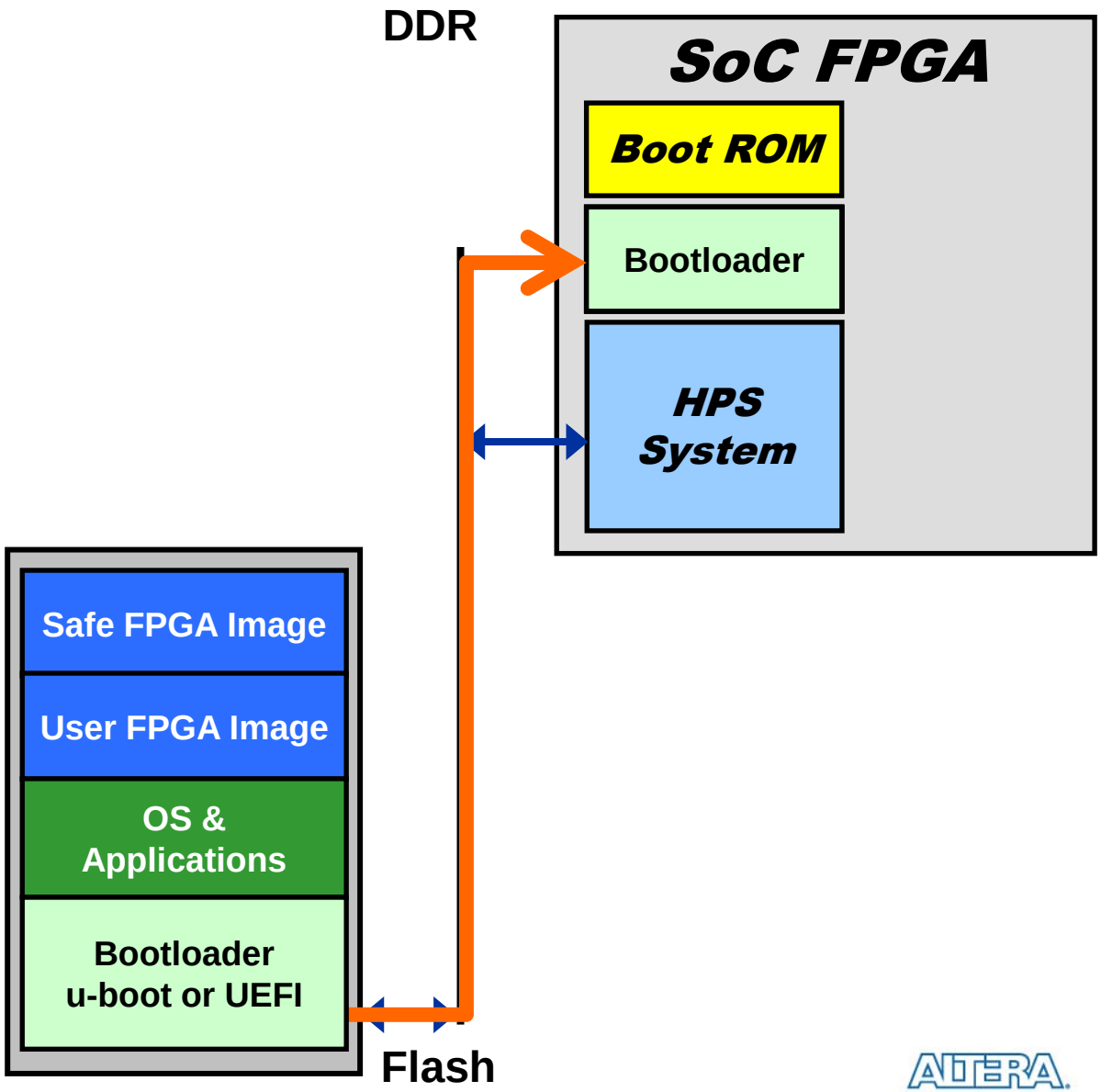
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins



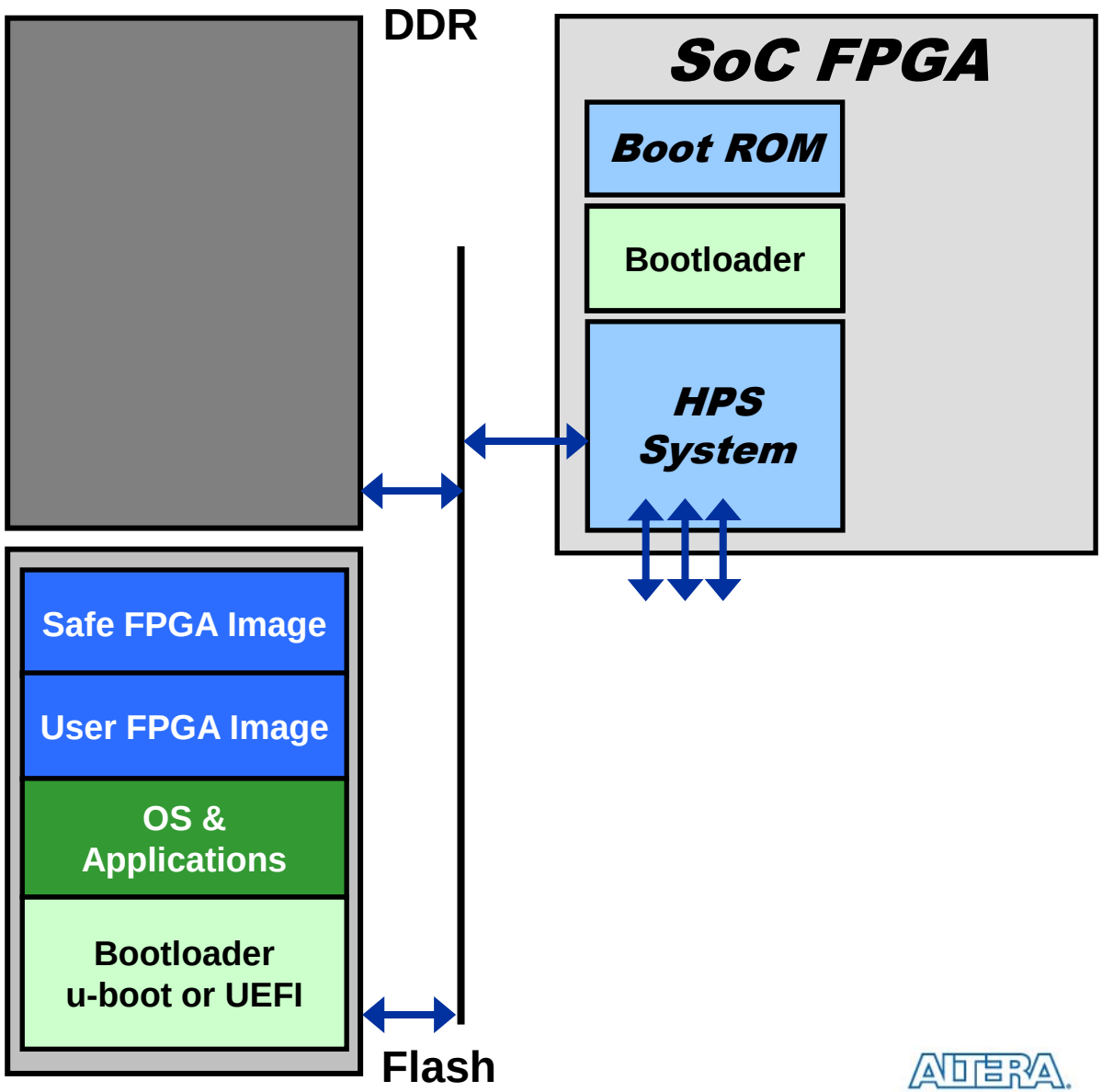
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM



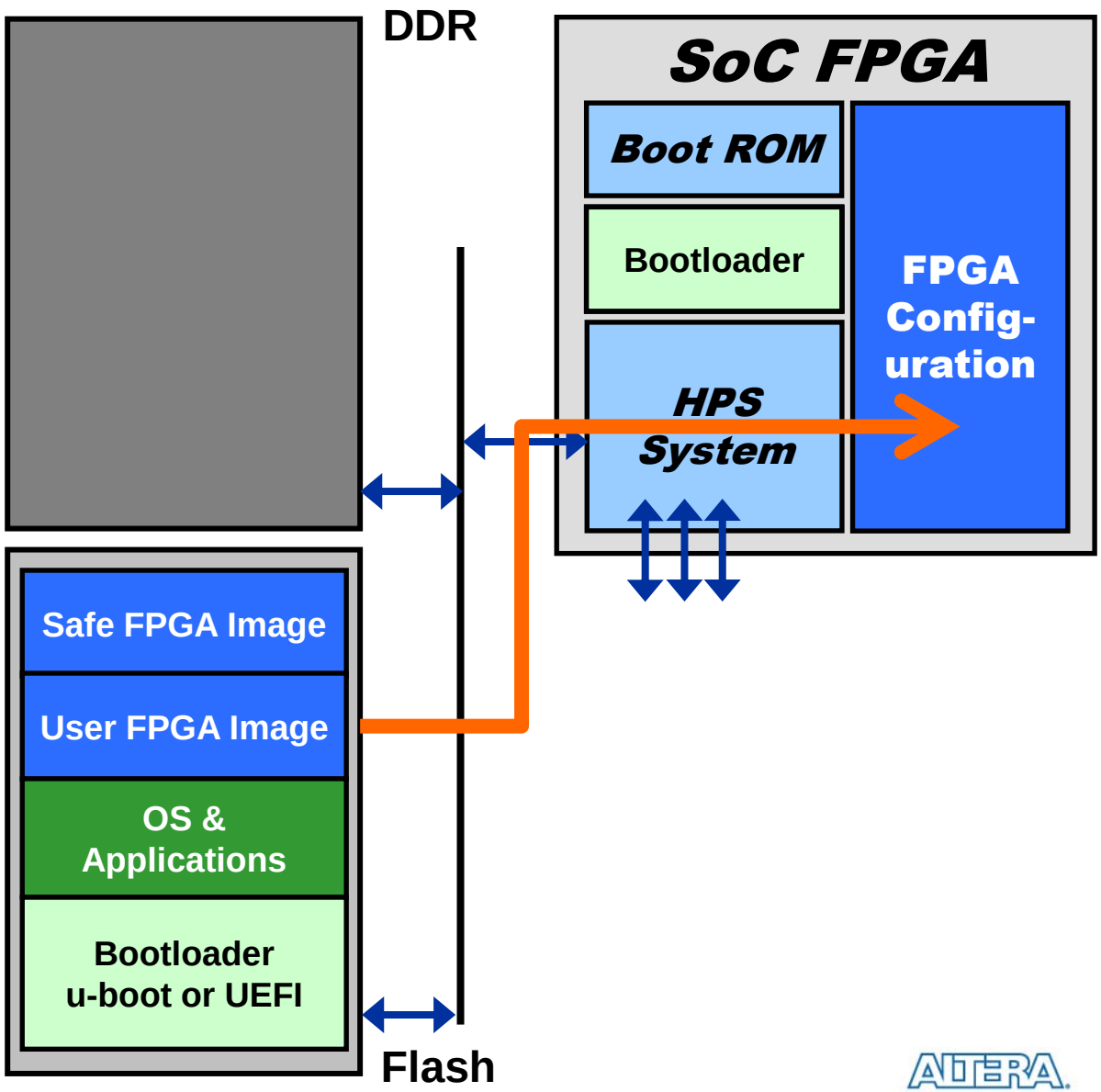
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM
5. Run Bootloader
6. Setup HPS I/O and DDR



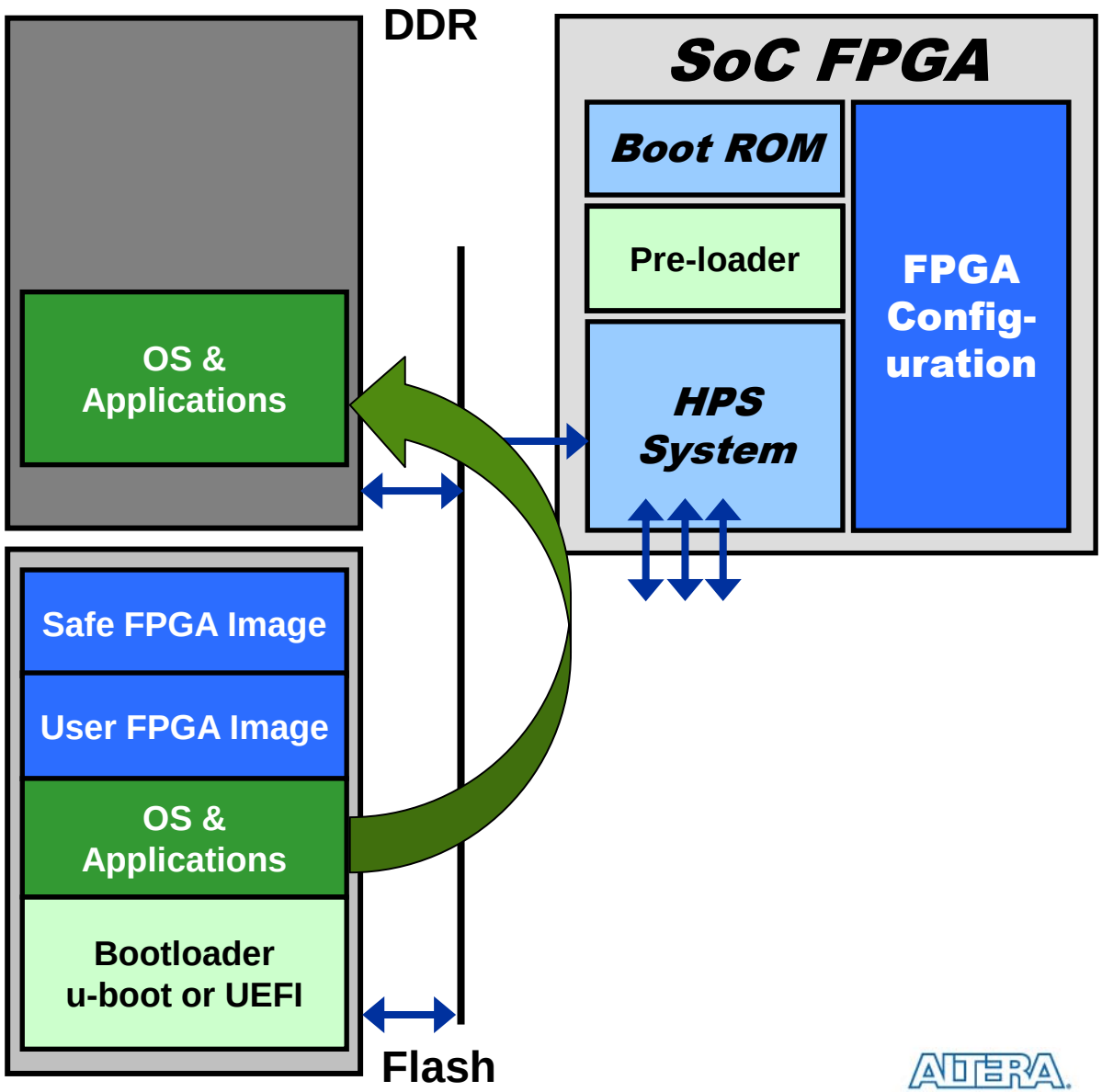
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM
5. Run Bootloader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)



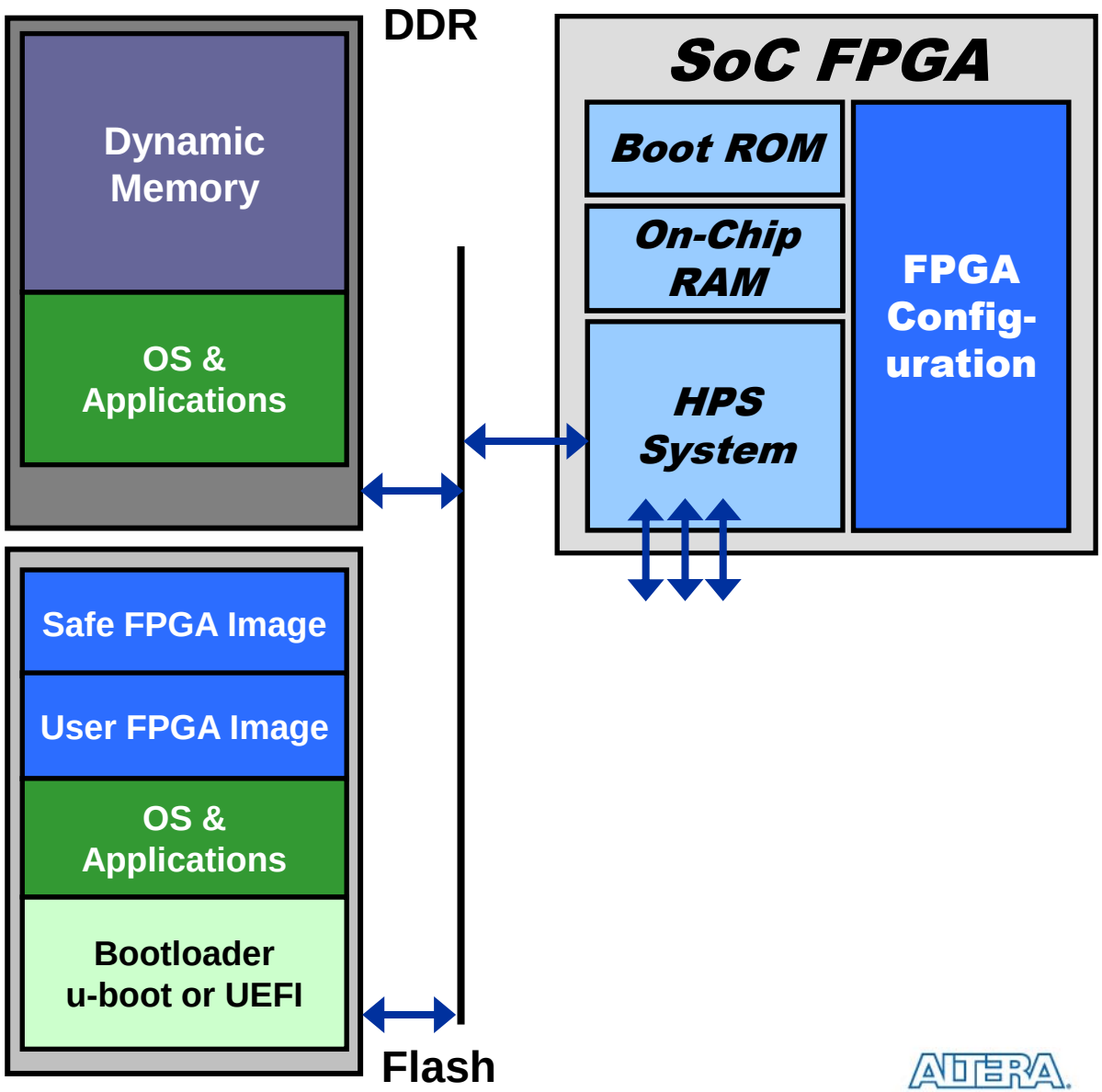
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM
5. Run Bootloader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy OS into DDR RAM



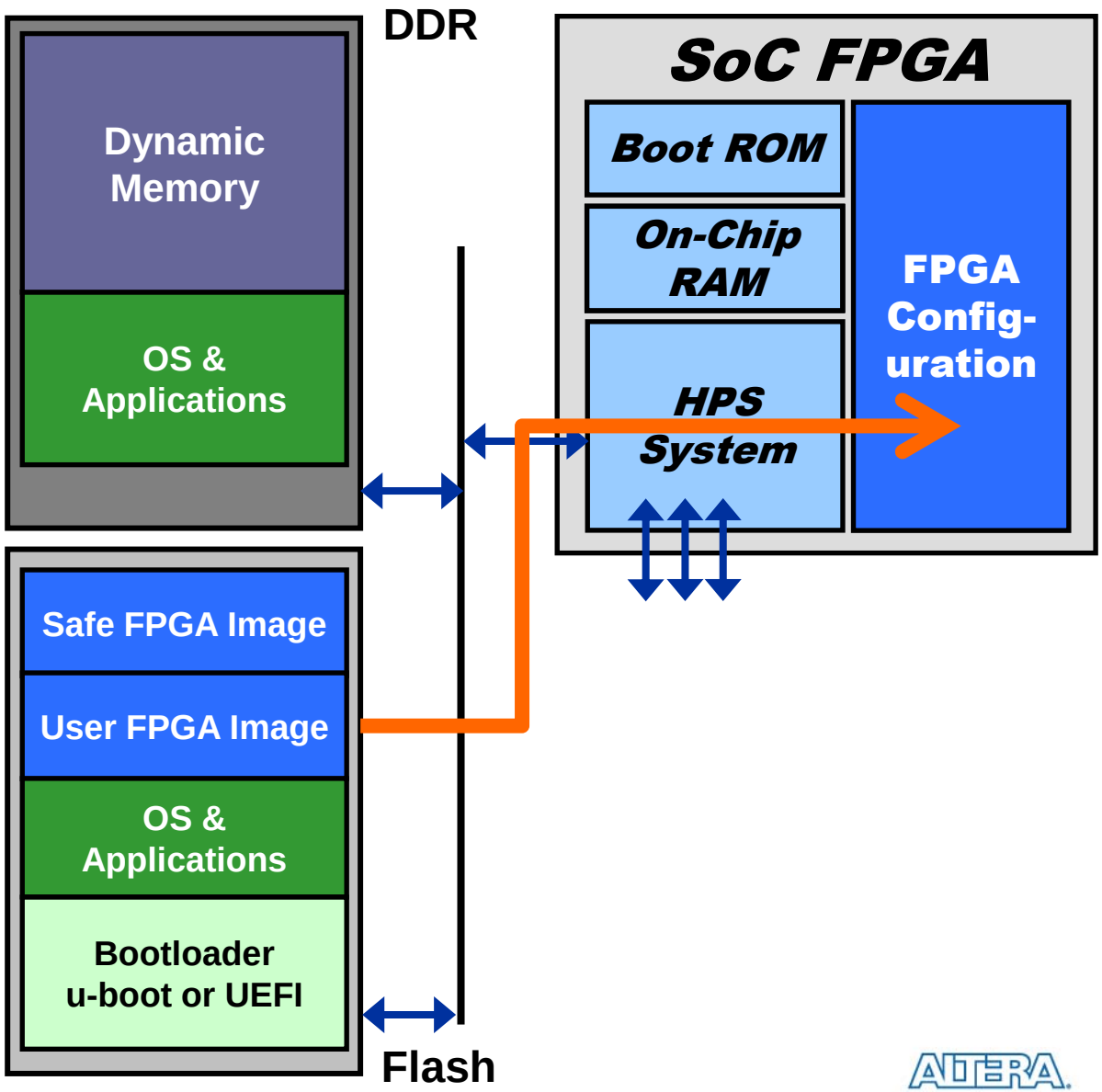
Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM
5. Run Bootloader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy OS into DDR RAM
9. Run OS
10. Run applications



Non-Secure Boot Stages – Arria 10

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Bootloader into Onchip RAM
5. Run Bootloader
6. Setup HPS I/O and DDR
7. Configure FPGA (optional)
8. Copy OS into DDR RAM
9. Run OS
10. Run applications
11. Configure FPGA (optional)



Learn more about boot and configuration process

- HPS SoC Boot Guide
  - Cyclone V SoC & Arria V SoC: AN709 - HPS SoC Boot Guide
  - Arria 10 SoC: included in HPS TRM in Arria 10 Device Handbook
    - Arria 10 SoC secure booting: AN759 – Arria 10 SoC Secure Boot User Guide
- Booting the HPS
  - Choose the device family you are interested in from the links below and then locate the “Hard Processor System Technical Reference Manual” for that device and locate the “Booting and Configuration” chapter in that document.
    - https://www.altera.com/products/soc/portfolio/cyclone-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-10-soc/support.html

Hardware Flows & Tools

Traditional System Development Flow

FPGA Design Flow

QUARTUS® II

Qsys

Hardware Development

Design	• Quartus II design software • Qsys system integration tool • Standard RTL flow • Altera and partner IP
Simulate	• ModelSim, VCS, NCSim, etc. • AMBA-AXI and Avalon bus functional models (BFMs)
Debug	• SignalTap™ II logic analyzer • System Console
Release	• Quartus II Programmer • In-system Update

Software Design Flow

Software Development

Design	• ARM Development Studio 5 • GNU toolchain • OS/BSP: Linux, VxWorks • Hardware Libraries • Design Examples
Simulate	• Virtual Platform
Debug	• GNU, Lauterbach, DS5
Release	• Flash Programmer

ALTERA now part of Intel

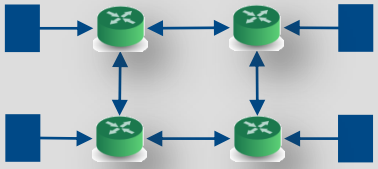
So… what exactly is Qsys?

GUI based system integration tool for HW system design using IP blocks.

- Simplifies complex system development
- Raises the level of design abstraction
- Provides a standard platform:
  - IP integration
  - Custom IP authoring
  - IP verification
- Enables design re-use
- Scales easily to meet the needs of end product
- Reduces time to market

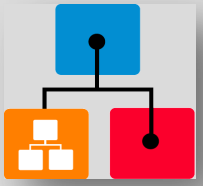
Qsys System Integration Platform

High-Performance Interconnect

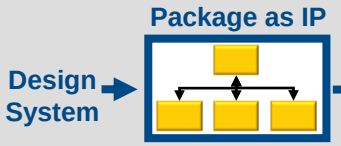


Based on Network-on-a-Chip (NoC) Architecture

Hierarchy



Design Reuse

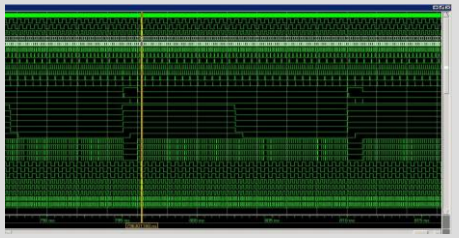


Qsys

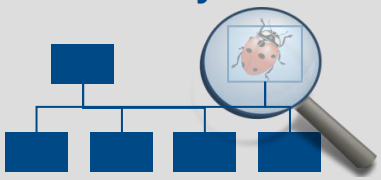
Industry-Standard Interfaces

ALTERA.	Avalon® Interfaces
ARM®	AMBA® AXI, APB, AHB

Automated Testbench Generation

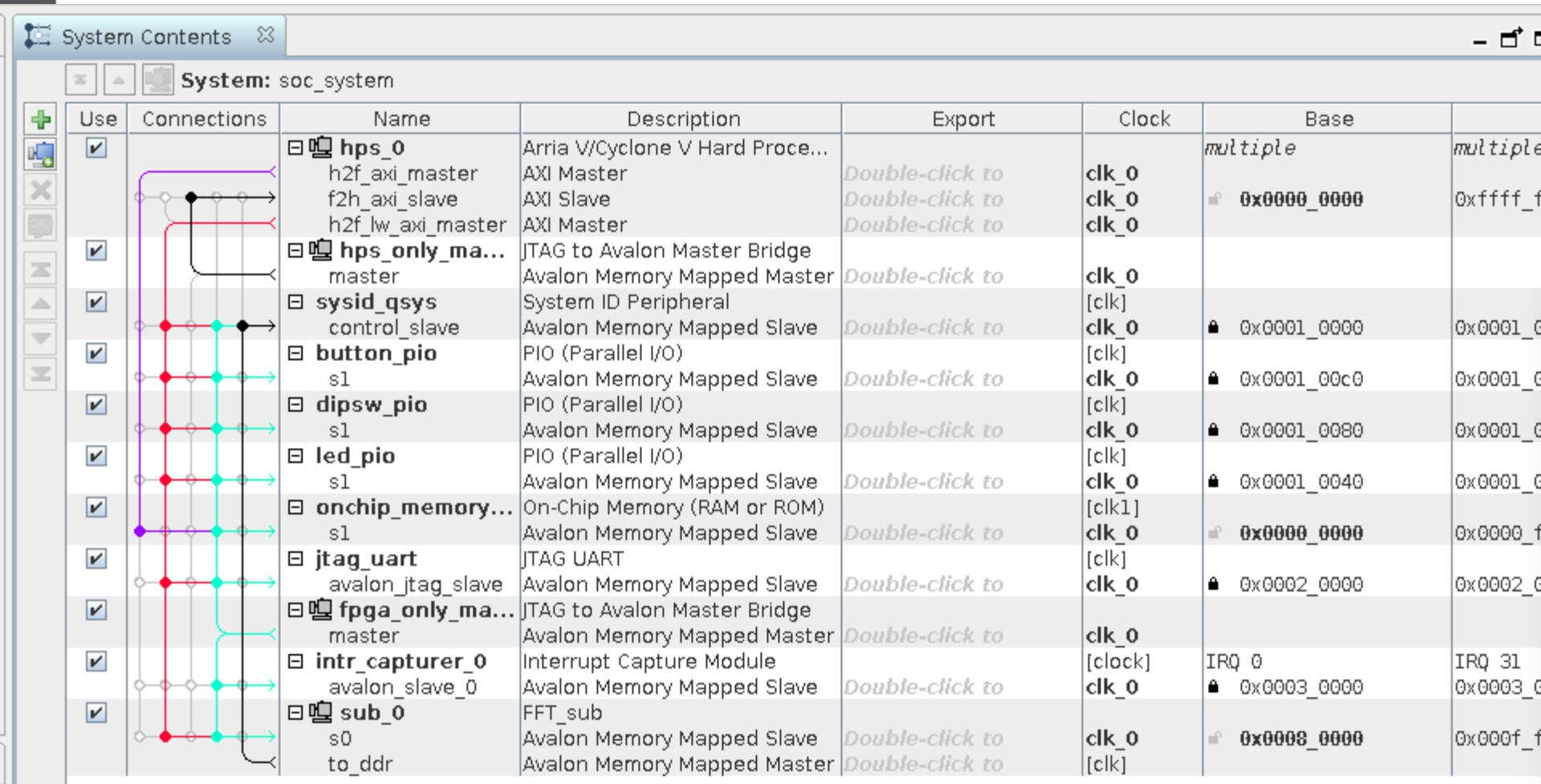


Real-Time System Debug



- Fastest way to build, modify, & optimize complex systems
- Flexibly optimize for performance or area
- Seamlessly integrate with Altera's embedded solutions

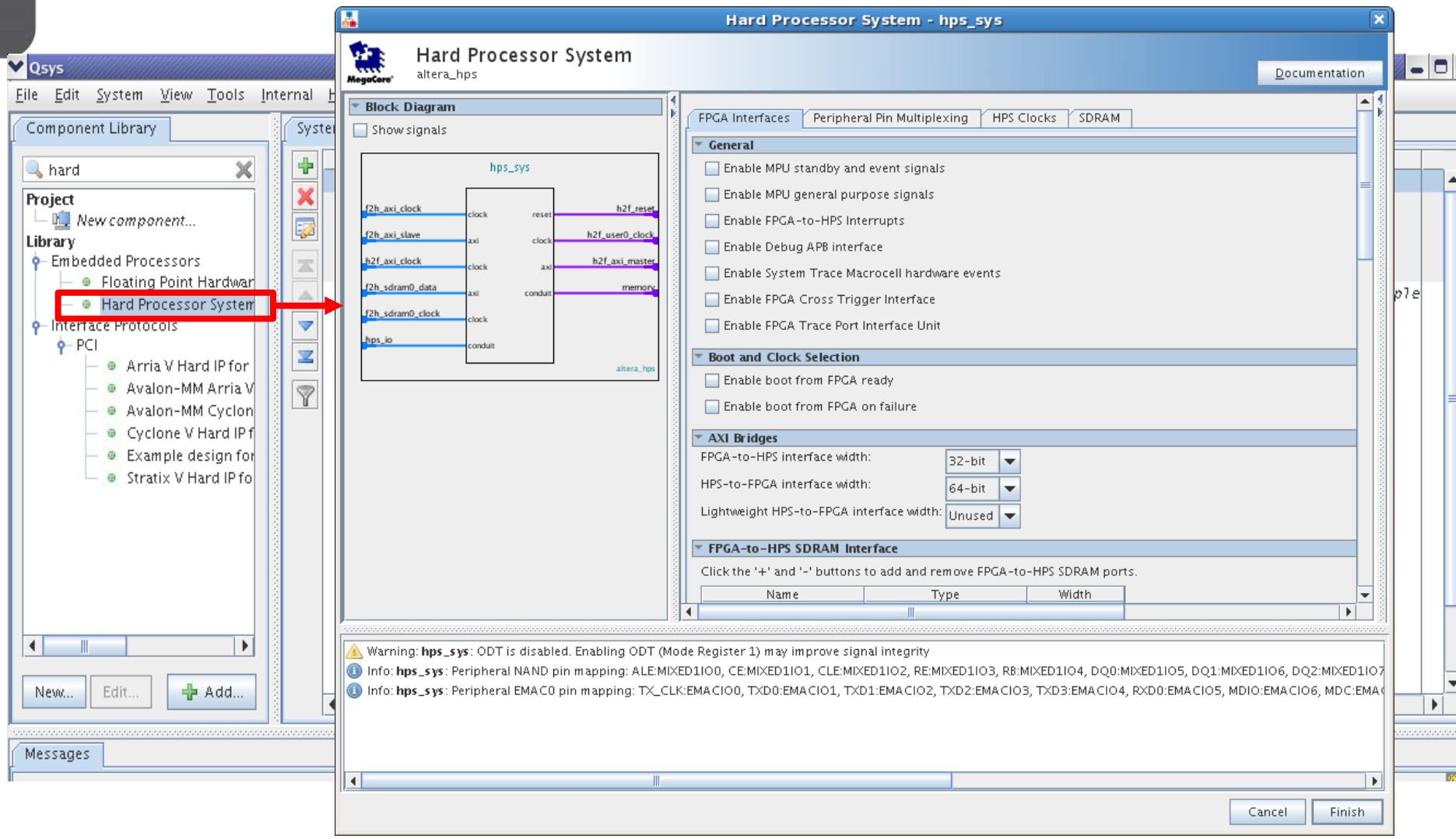
A GHRD Qsys system



Use	Connections	Name	Description	Export	Clock	Base	
☑		⊟ hps_0	Arria V/Cyclone V Hard Proce...			*multiple*	*multiple*
		h2f_axi_master	AXI Master	Double-click to	clk_0		
		f2h_axi_slave	AXI Slave	Double-click to	clk_0	0x0000_0000	0xffff_f
		h2f_lw_axi_master	AXI Master	Double-click to	clk_0		
☑		⊟ hps_only_ma...	JTAG to Avalon Master Bridge				
		master	Avalon Memory Mapped Master	Double-click to	clk_0		
☑		⊟ sysid_qsys	System ID Peripheral		[clk]		
		control_slave	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0001_0000	0x0001_0
☑		⊟ button_pio	PIO (Parallel I/O)		[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0001_00c0	0x0001_0
☑		⊟ dipsw_pio	PIO (Parallel I/O)		[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0001_0080	0x0001_0
☑		⊟ led_pio	PIO (Parallel I/O)		[clk]		
		s1	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0001_0040	0x0001_0
☑		⊟ onchip_memory...	On-Chip Memory (RAM or ROM)		[clk1]		
		s1	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0000_0000	0x0000_f
☑		⊟ jtag_uart	JTAG UART		[clk]		
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0002_0000	0x0002_0
☑		⊟ fpga_only_ma...	JTAG to Avalon Master Bridge				
		master	Avalon Memory Mapped Master	Double-click to	clk_0		
☑		⊟ intr_capturer_0	Interrupt Capture Module		[clock]	IRQ 0	IRQ 31
		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0003_0000	0x0003_0
☑		⊟ sub_0	FFT_sub				
		s0	Avalon Memory Mapped Slave	Double-click to	clk_0	0x0008_0000	0x000f_f
		to_ddr	Avalon Memory Mapped Master	Double-click to	[clk]		

- Qsys connects masters and slaves together
- Configures components
- Defines memory maps

Hard Processor System Component



Learn more Qsys and HPS configuration

- Avalon interface specification
  - http://www.altera.com/literature/manual/mnl_avalon_spec.pdf
- Qsys resource page
  - http://www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qts-qsys.html
- Configuring the HPS in Qsys
  - Choose the device family you are interested in from the links below and then locate the “Hard Processor System Technical Reference Manual” for that device and locate the “Instantiating the HPS Component” chapter in that document.
    - https://www.altera.com/products/soc/portfolio/cyclone-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-v-soc/support.html
    - https://www.altera.com/products/soc/portfolio/arria-10-soc/support.html

Software Flows and Tools

Traditional System Development Flow

FPGA Design Flow

Software Design Flow

Hardware Development

Software Development

- Quartus II design software
- Qsys system integration tool
- Standard RTL flow
- Altera and partner IP

Design

HW/SW Handoff

Design

- ARM Development Studio 5
- GNU toolchain
- OS/BSP: Linux, VxWorks
- Hardware Libraries
- Design Examples

- ModelSim, VCS, NCSim, etc.
- AMBA-AXI and Avalon bus functional models (BFMs)

Simulate

Simulate

- Virtual [illegible]

Software Development

- SignalTap™ II logic analyzer
- System Console

Debug

Debug

- GNU, Lauterbach, DS5

FPGA-Adaptive Debugging

- Quartus II Programmer
- In-system Update

Release

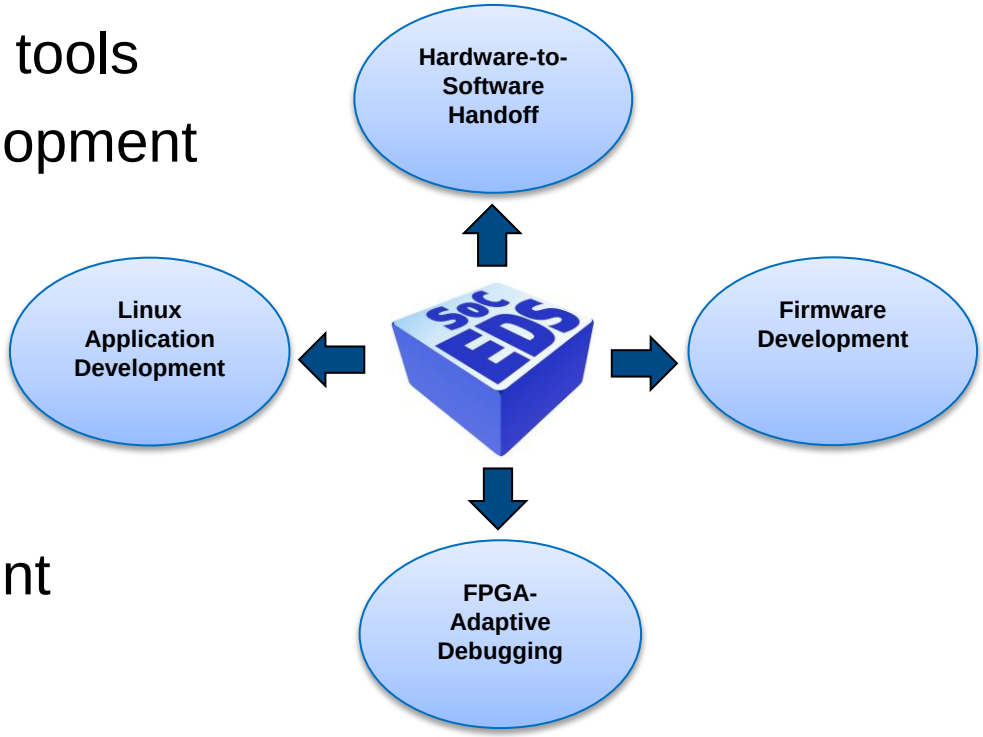
Release

- Flash Programmer

Altera SoC Embedded Design Suite

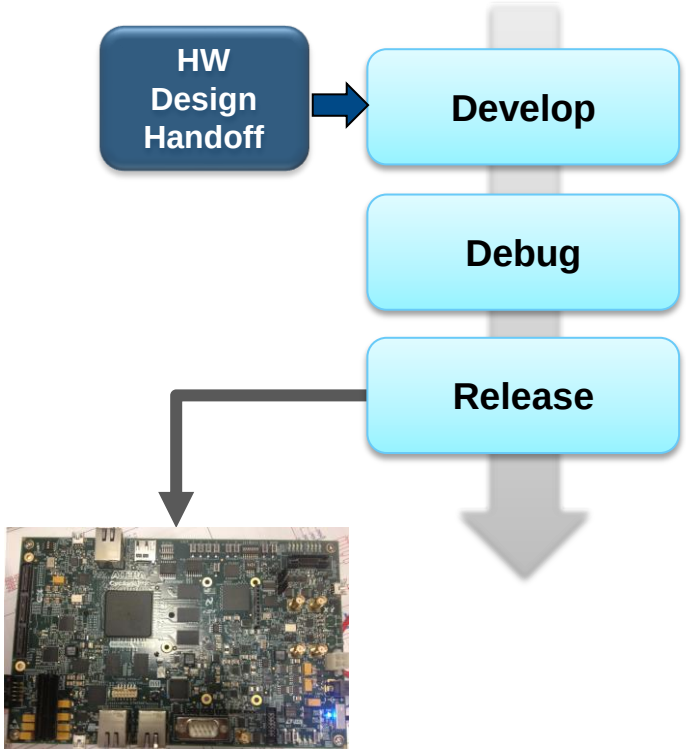
Comprehensive Suite SW Dev Tools

- Hardware / software handoff tools
- Bare-metal application development
  - SoC Hardware Libraries
  - Bare-metal compiler tools
- FPGA-adaptive debugging
  - ARM DS-5 Altera Edition Toolkit
- Linux application development
  - Yocto Linux build environment
  - Pre-built binaries for Linux / u-boot
  - Work in conjunction with the Community Portal – www.Rocketboards.org
- Design examples



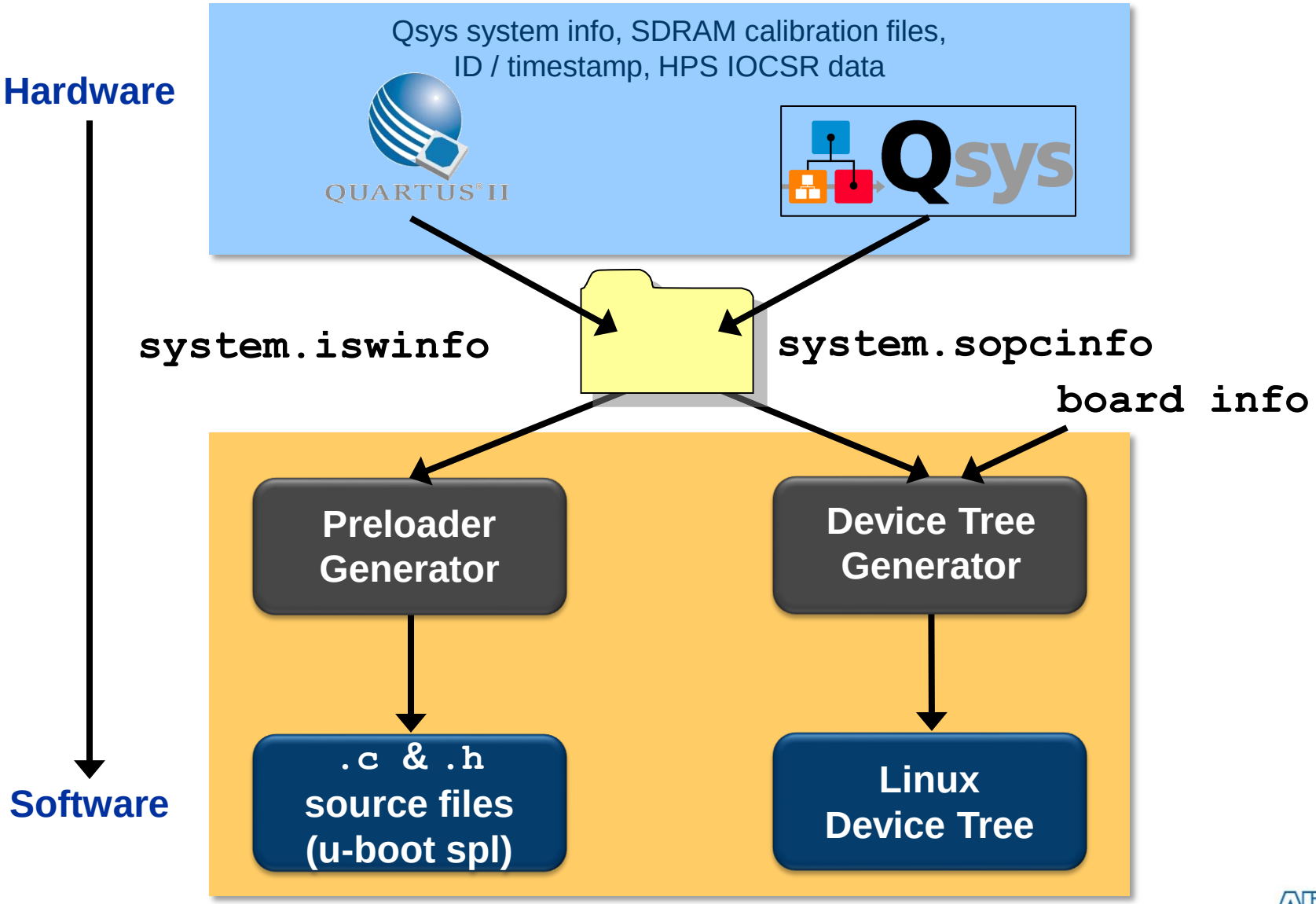
Software Design Flow

- Standard development environment
  - ARM DS-5 and/or partner IDE
- Standard software enablement
  - HWLIBs for use with or without OS
- Standard design flow
  - No proprietary or additional tools required

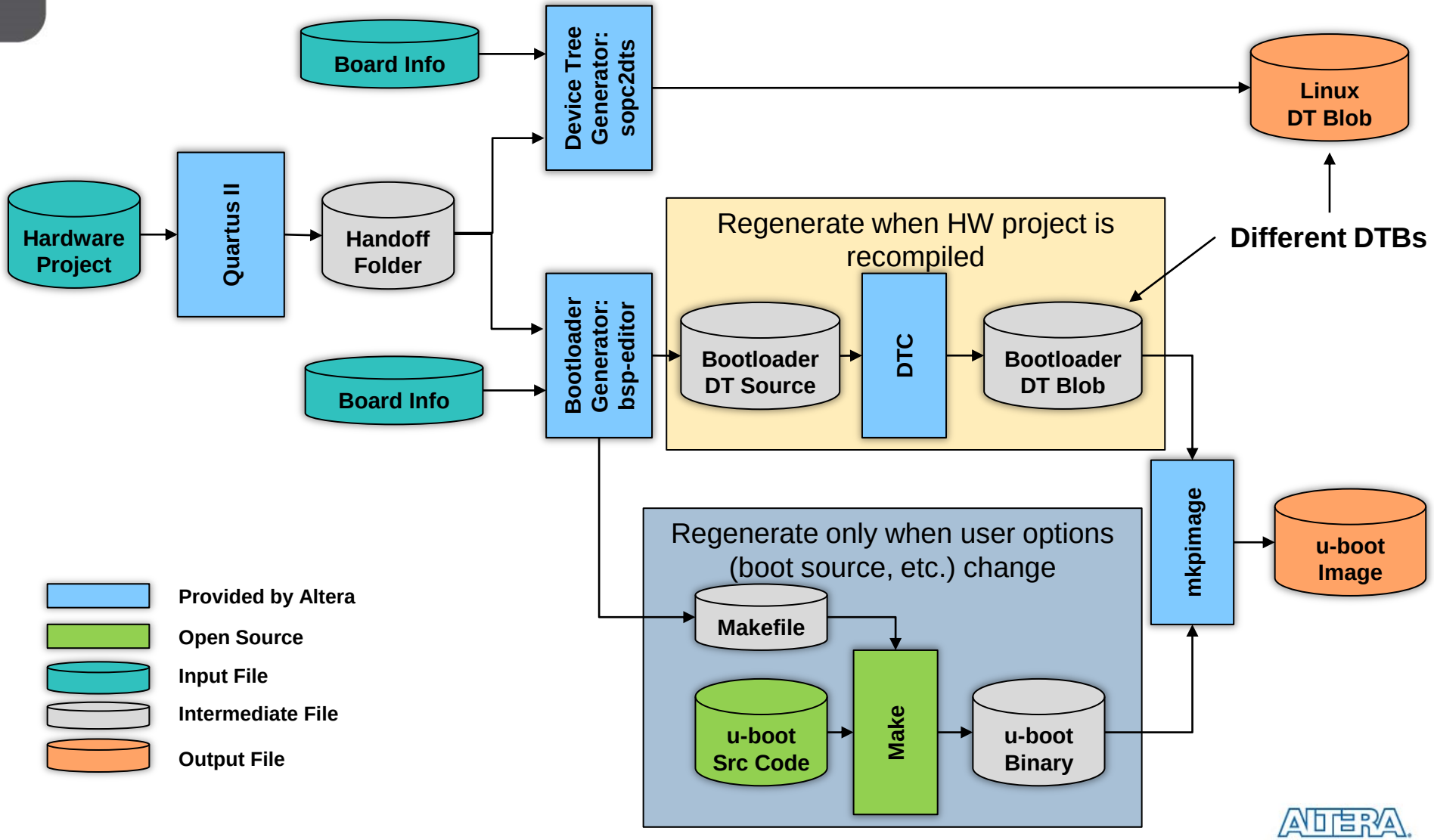


Altera SoCs provide high SW developer productivity

Linux HW/SW Handoff – Cyclone V SoC and Arria V SoC



Linux HW/SW Handoff – Arria 10 SoC

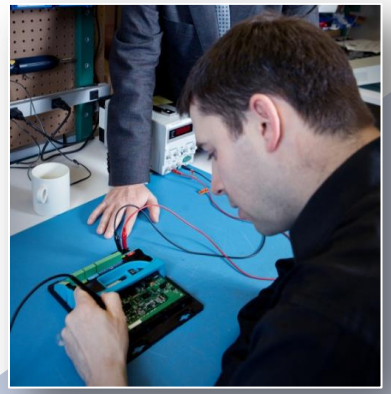


DS-5 Altera Edition

ARM DS-5 Altera Edition Overview

- ARM-Altera strategic partnership with unique OEM arrangement
- Low cost and included in many SoC development kits
- Complete multi-core debug and ARM CoreSight compliant trace
- Includes ARM Pro bare-metal compiler and Altera GCC compiler
- Industry-only FPGA-Adaptive debug support
- Uses USB-Blaster cable

DS-5 Altera Edition- One Tool, Three Usages



SoC EDS

Qsys

SignalTap II

1

- JTAG-Based Debugging
  - Board Bring-up
  - OS porting, Drivers Dev,
  - Kernel Debug

2

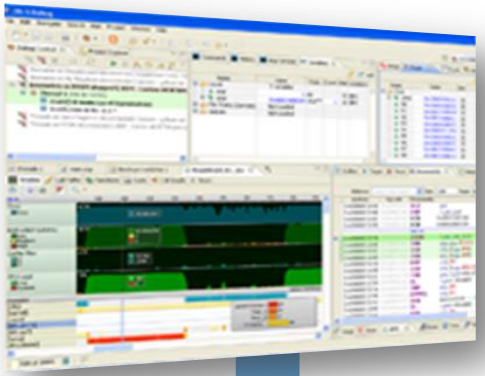
- Application Debugging
  - Linux User Space Code
  - RTOS App Code

3

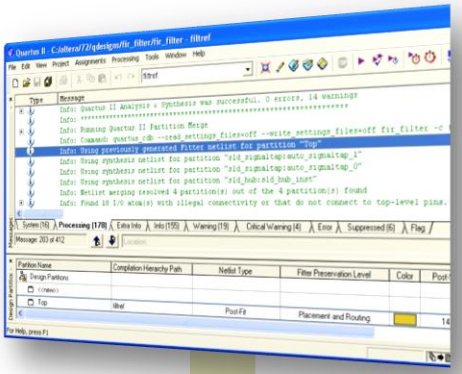
- FPGA-Adaptive Debugging
  - System Integration
  - System Debug

One Device, Two Debugging Tools?

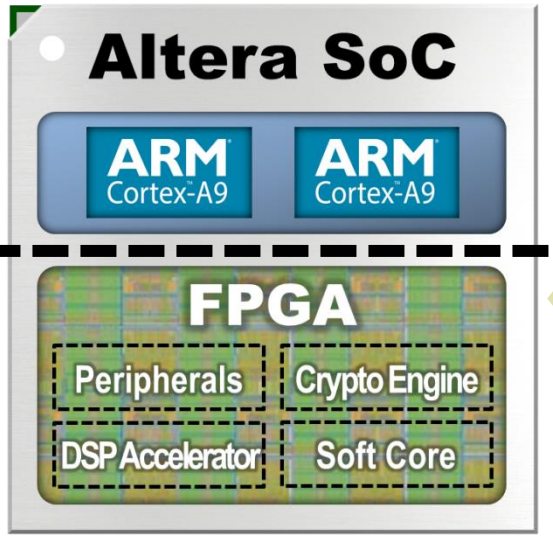
ARM® DS-5™ Toolkit



Altera Quartus™ II Software



JTAG

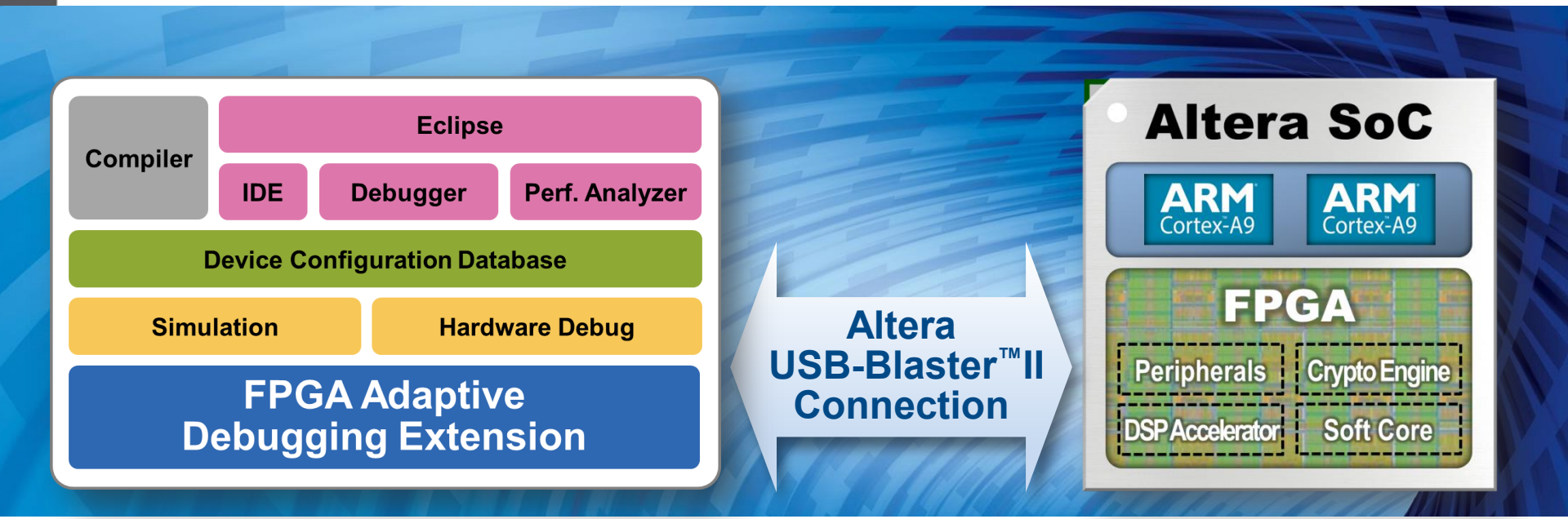


JTAG

- Dedicated JTAG connection
- Visualize & control CPU subsystem

- Dedicated JTAG connection
- Visualize & control FPGA

Industry First: FPGA-Adaptive Debugging

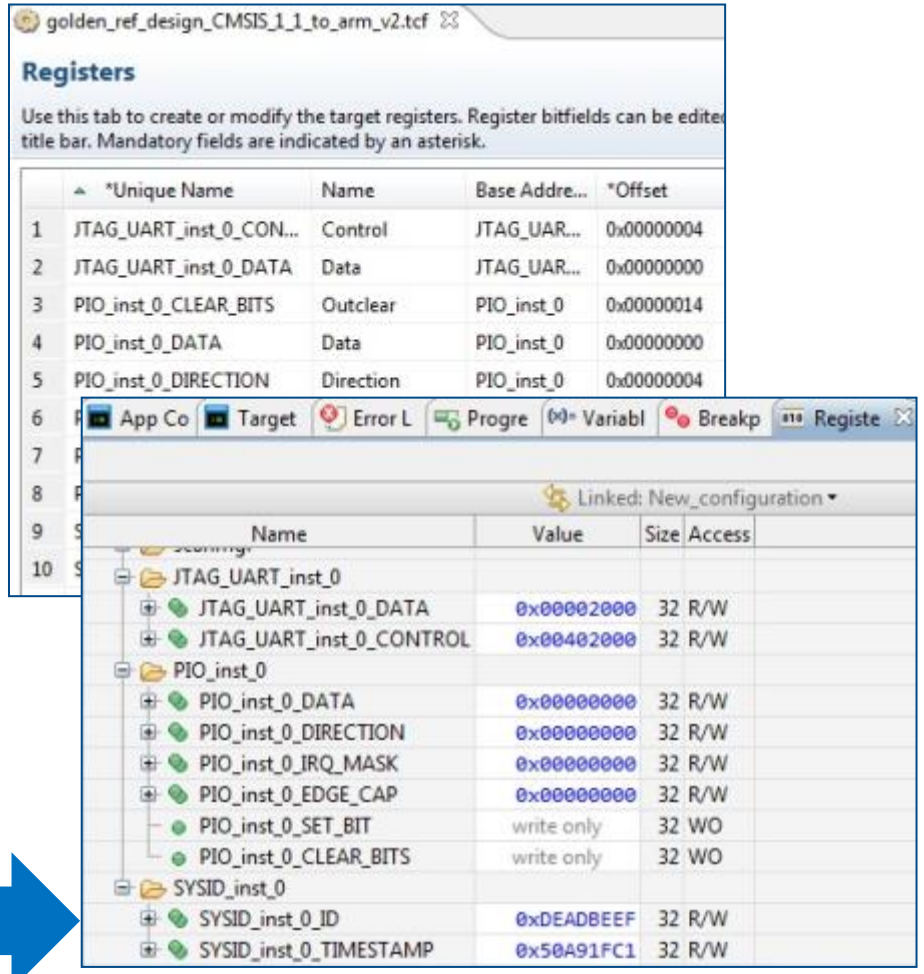


ARM® Development Studio 5 (DS-5™) Altera® Edition Toolkit

- Removes debugging barrier between CPUs and FPGA
- Exclusive OEM agreement between Altera and ARM
- Result of innovation in silicon, software, and business model

Visualization of SoC Peripherals

- Register views assist the debug of FPGA peripherals
  - File generated by FPGA tool flow
  - Automatically imported in DS-5 Debugger
- Debug views for debug of software drivers
  - Self-documenting
  - Grouped by peripheral, register and bit-field

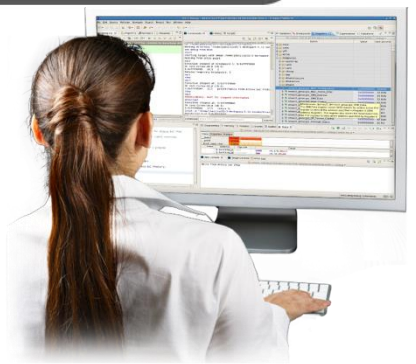


Qsys

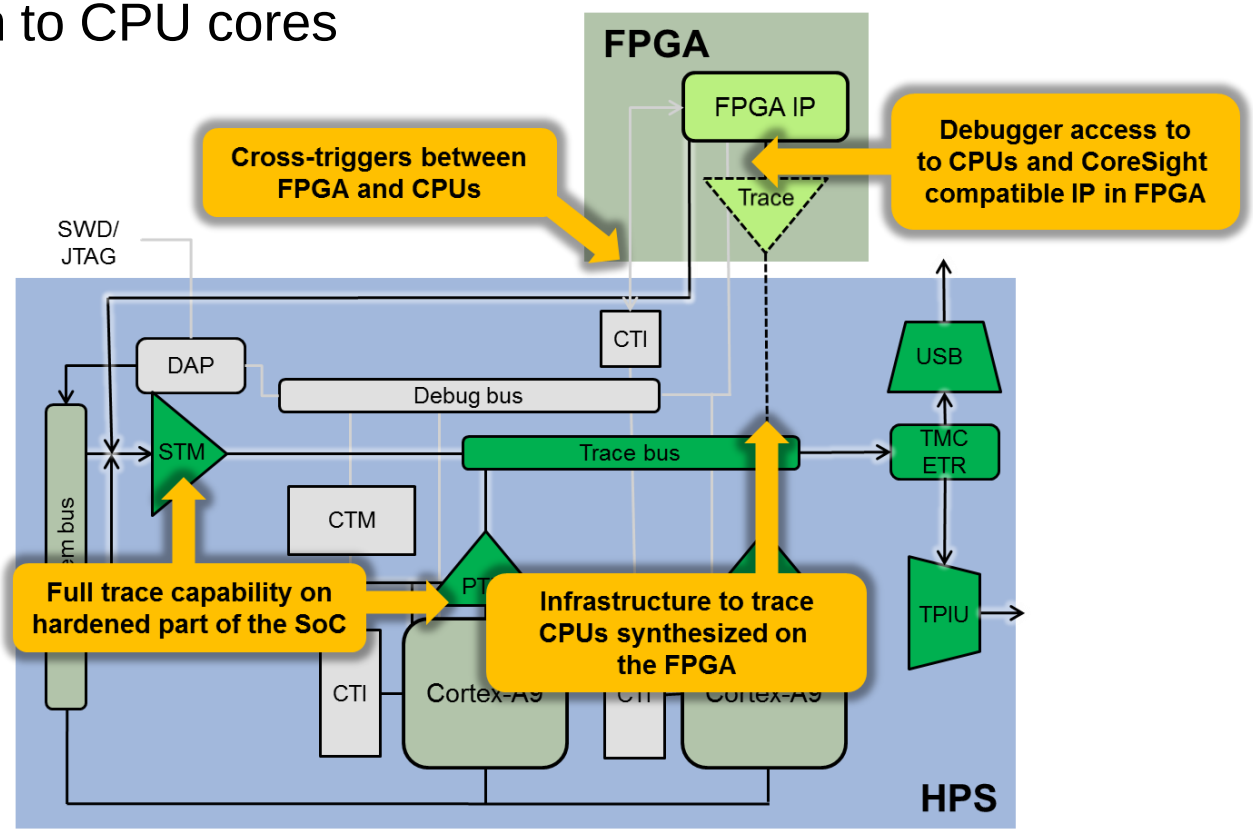
CMSIS

Peripheral register descriptions

FPGA-Adaptive, Unified Debugging

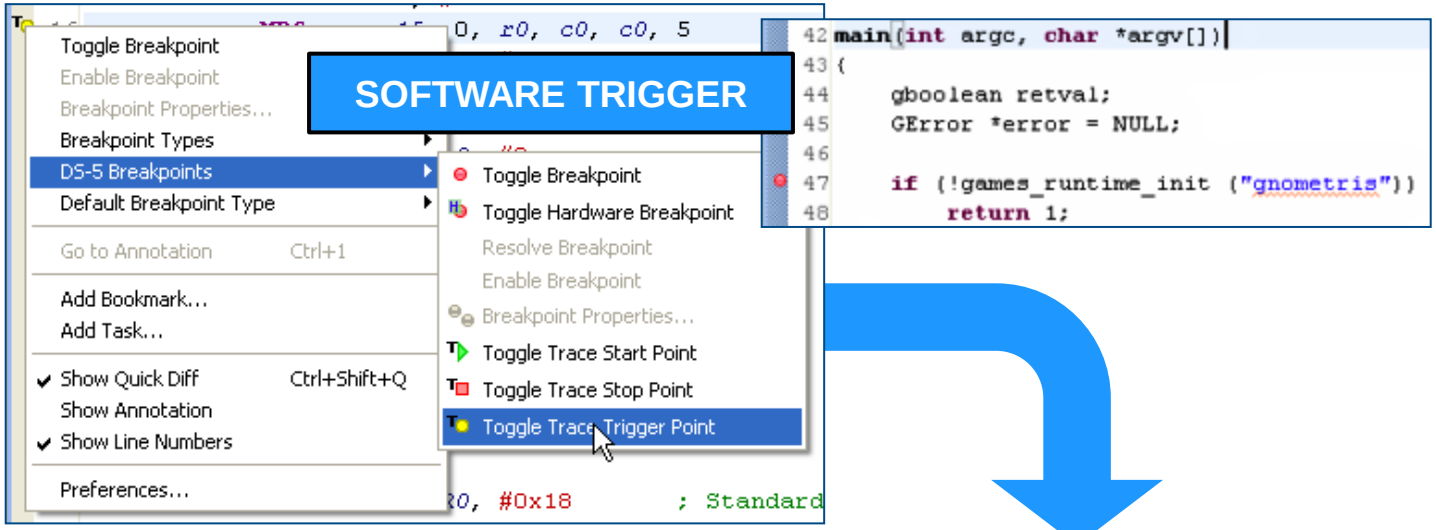


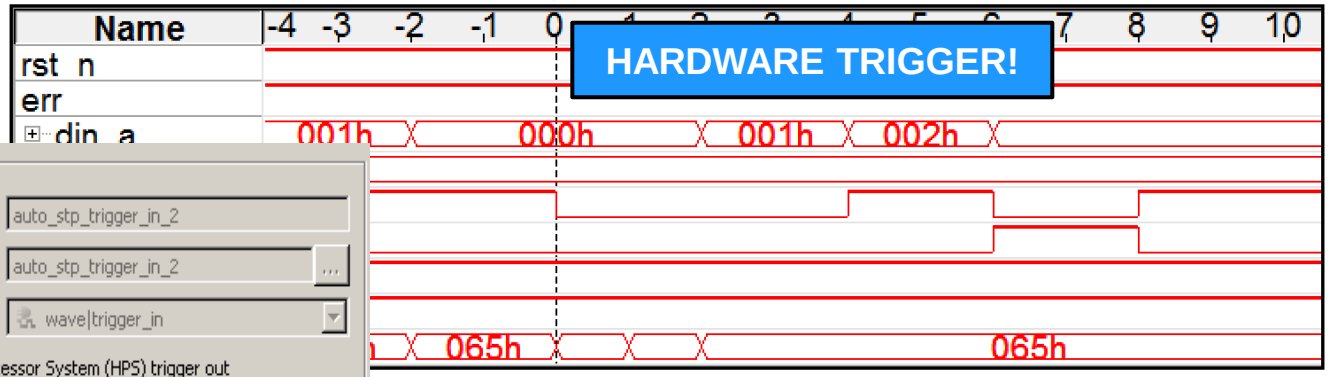
- FPGA connected to debug and trace buses for non-intrusive capture and visualization of signal events
- Simultaneous debug and trace connection to CPU cores and compatible IP
- Correlate FPGA signal events with software events and CPU instruction trace using triggers and timestamps



Cross-Domain Debug 1

- Trigger from software world to FPGA world

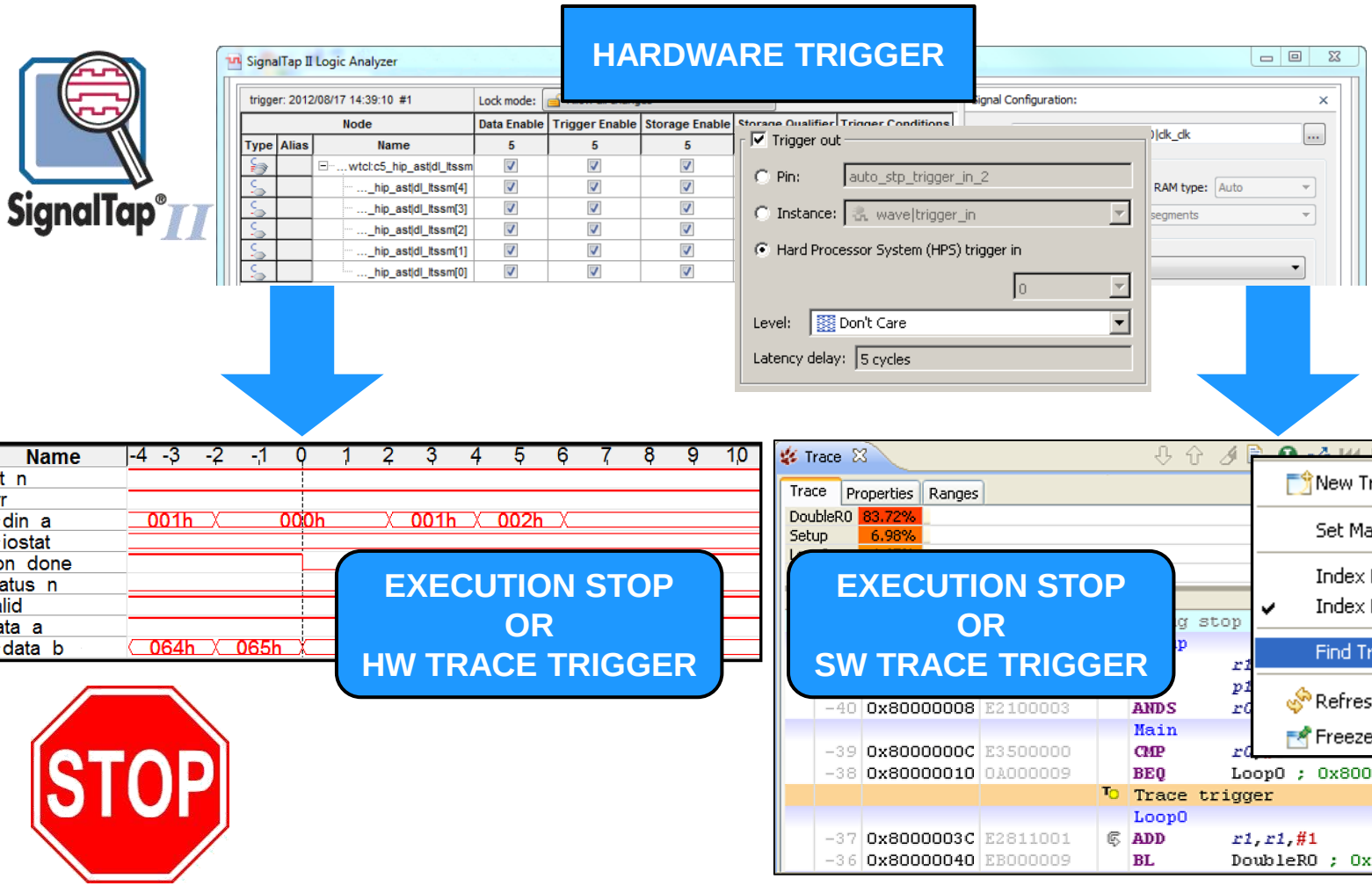




SignalTap II

Cross-Domain Debug 2

- Trigger from FPGA world to software world



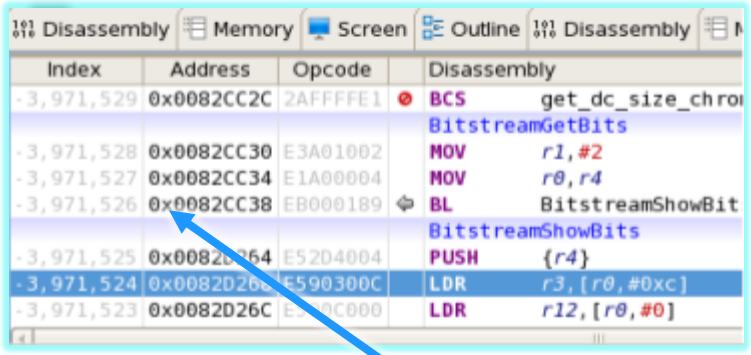
Correlate HW and SW Events

- Debug event trigger point set from either:

  SignalTap™ II Logic Analyzer
  or
  DS-5 debugger

- Captured trace can then be analyzed using timestamp-correlated events

ARM® DS-5™ Toolkit



Timestamp Correlated

SignalTap II Logic Analyzer

SoC EDS Summary with DS-5 Editions Summary

Component	Key Feature	Web Edition	Subscription Edition	30-Day Evaluation
Hardware/Software Handoff Tools	Preloader Image Generator	x	x	x
	Flash Image Creator	x	x	x
	Device Tree Generator (Linux)	x	x	x
ARM DS-5 Altera Edition	Eclipse IDE	x	x	x
	Debugging over Ethernet (Linux)	x	x	x
	Debugging over USB-Blaster II JTAG		x	x
	Automatic FPGA Register Views		x	x
	Hardware Cross-triggering		x	x
	CPU/FPGA Event Correlation		x	x
Compiler Tool Chains	Linaro Tool Chain (Linux)	x	x	x
	GCC EABI (Bare-metal)	X	x	x
	ARM Pro Compiler		x	
Hardware Libraries	Bare-metal programming Support	x	x	x
SoC Programming Examples	Golden System Reference Design	x	x	x

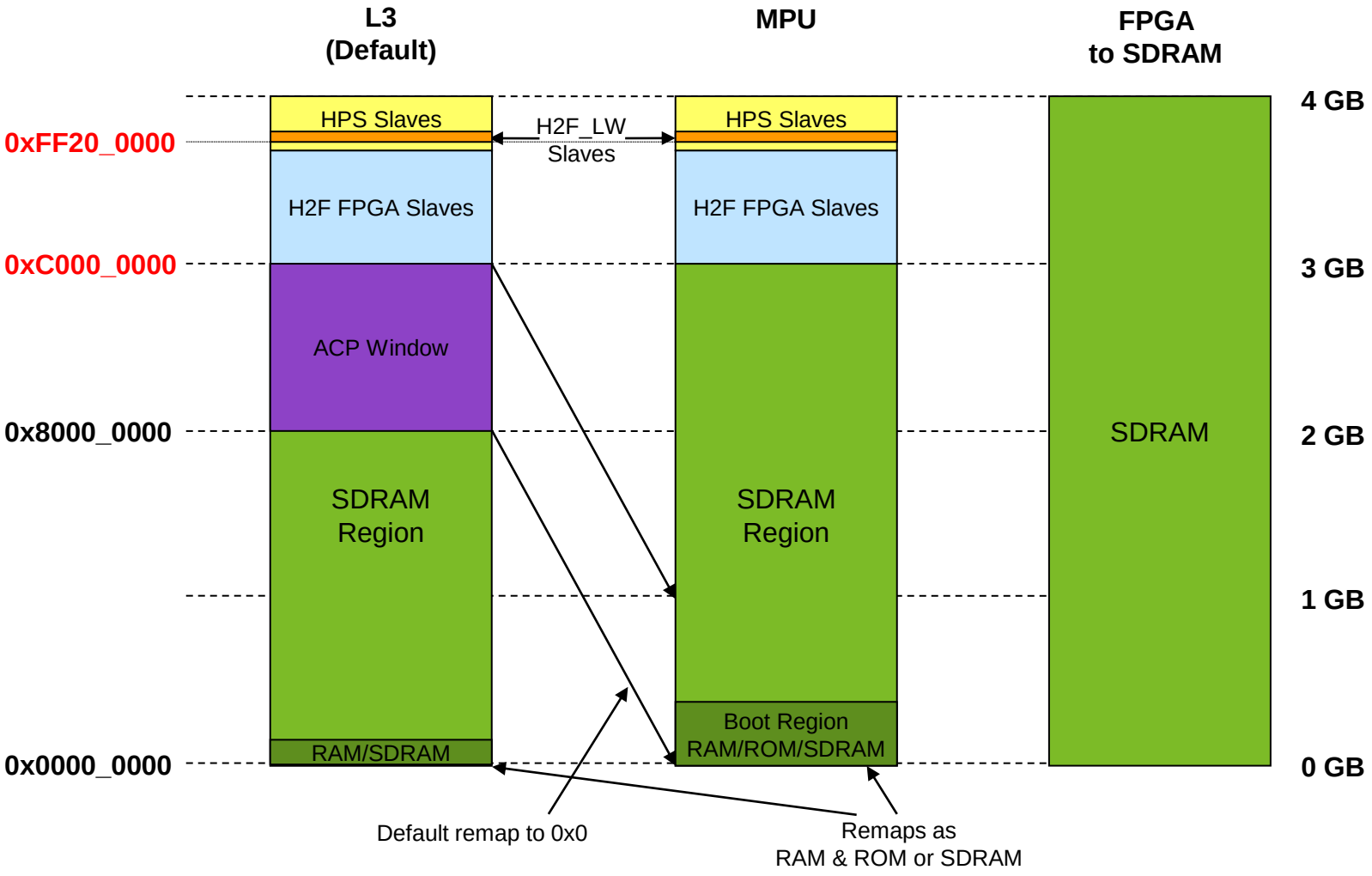
Everything needed for Linux application development is free

Learn more about DS-5

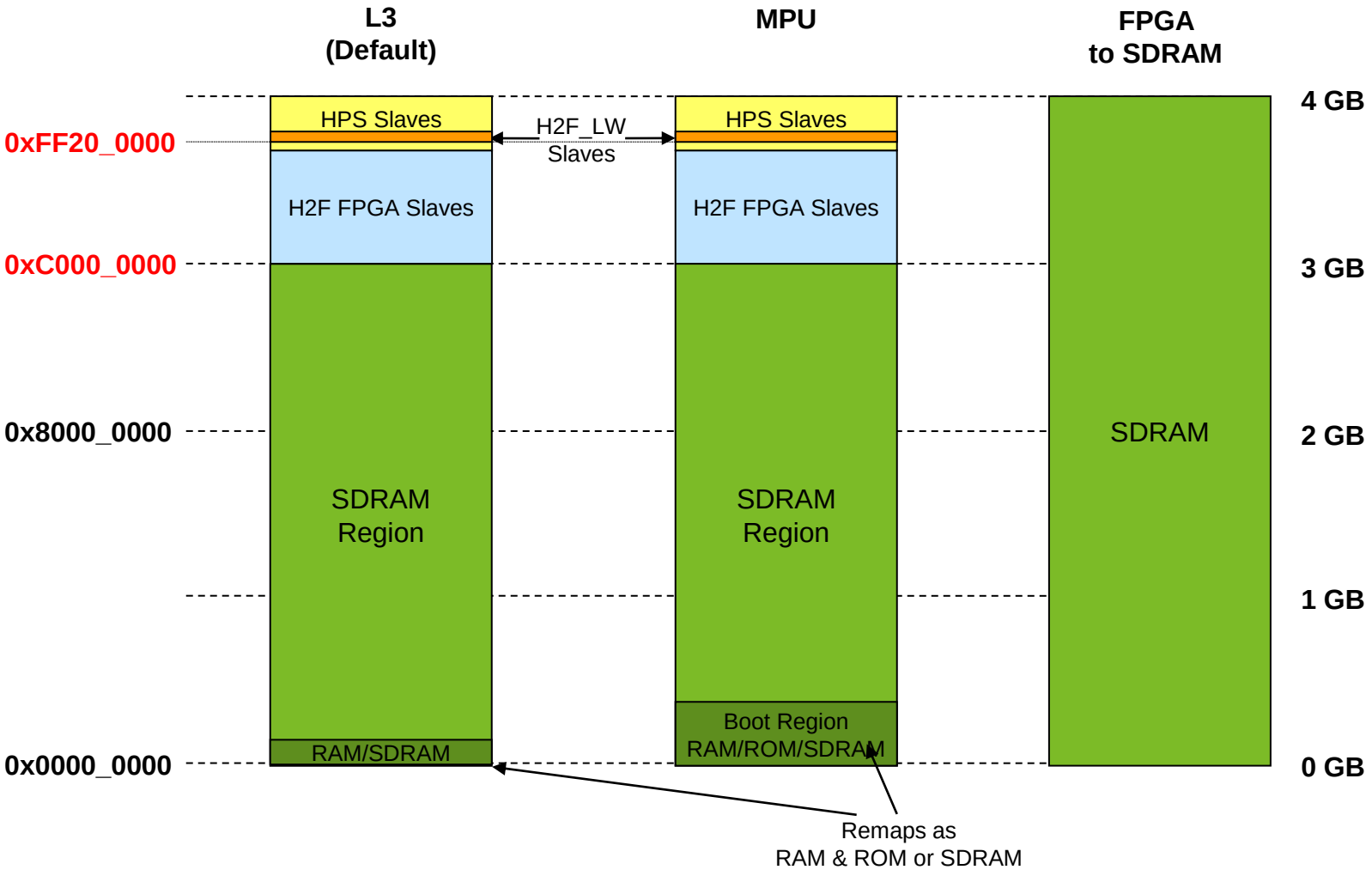
- ARM
  - http://ds.arm.com
- Altera Edition
  - http://www.altera.com/devices/processor/arm/cortex-a9/software/proc-arm-development-suite-5.html
- SoC EDS user guide
  - https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_soc_eds.pdf

SoC Physical Address Map

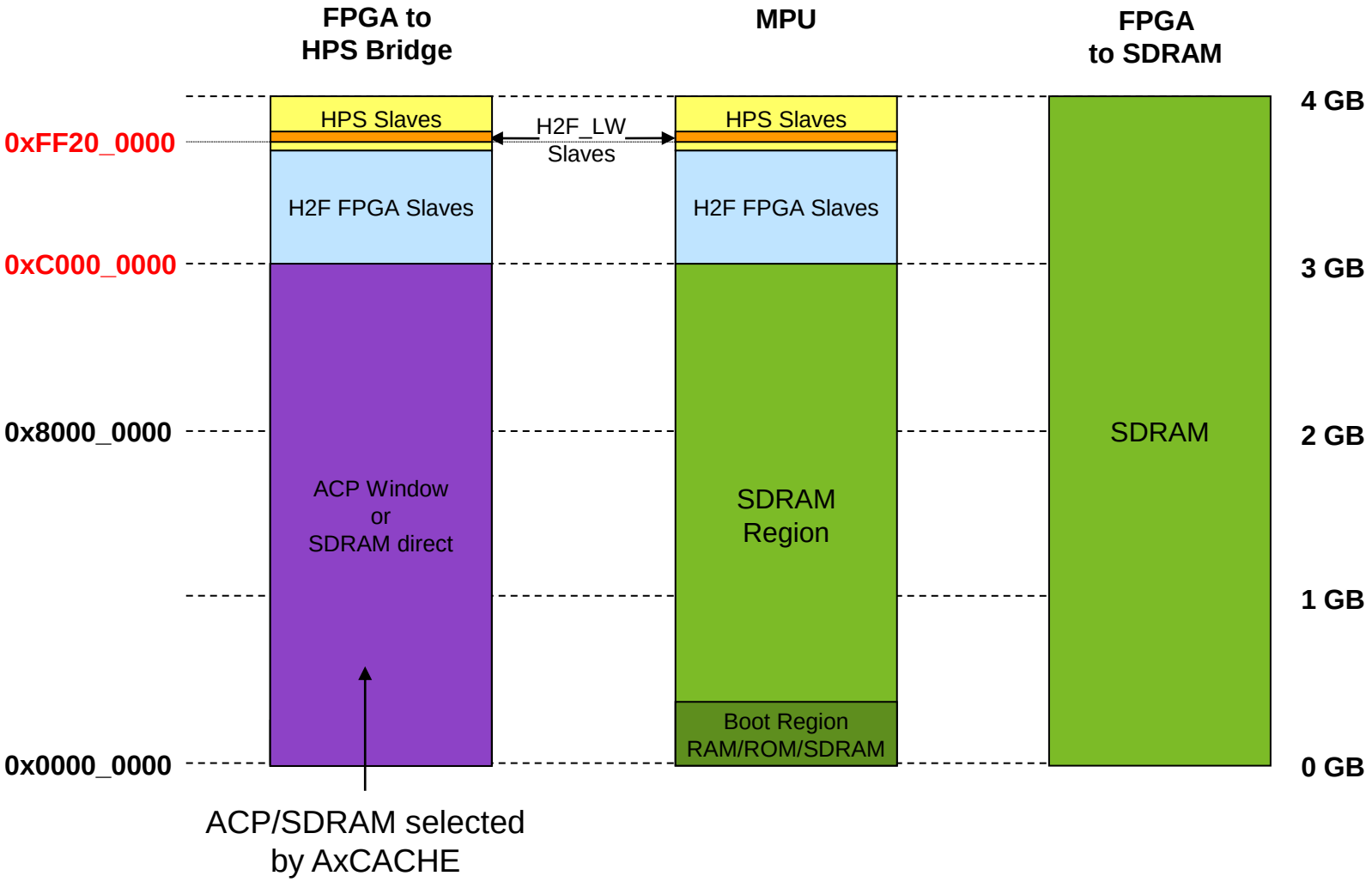
Cyclone V & Arria V SoC HPS Physical Memory Map



Arria 10 SoC HPS Physical Memory Map

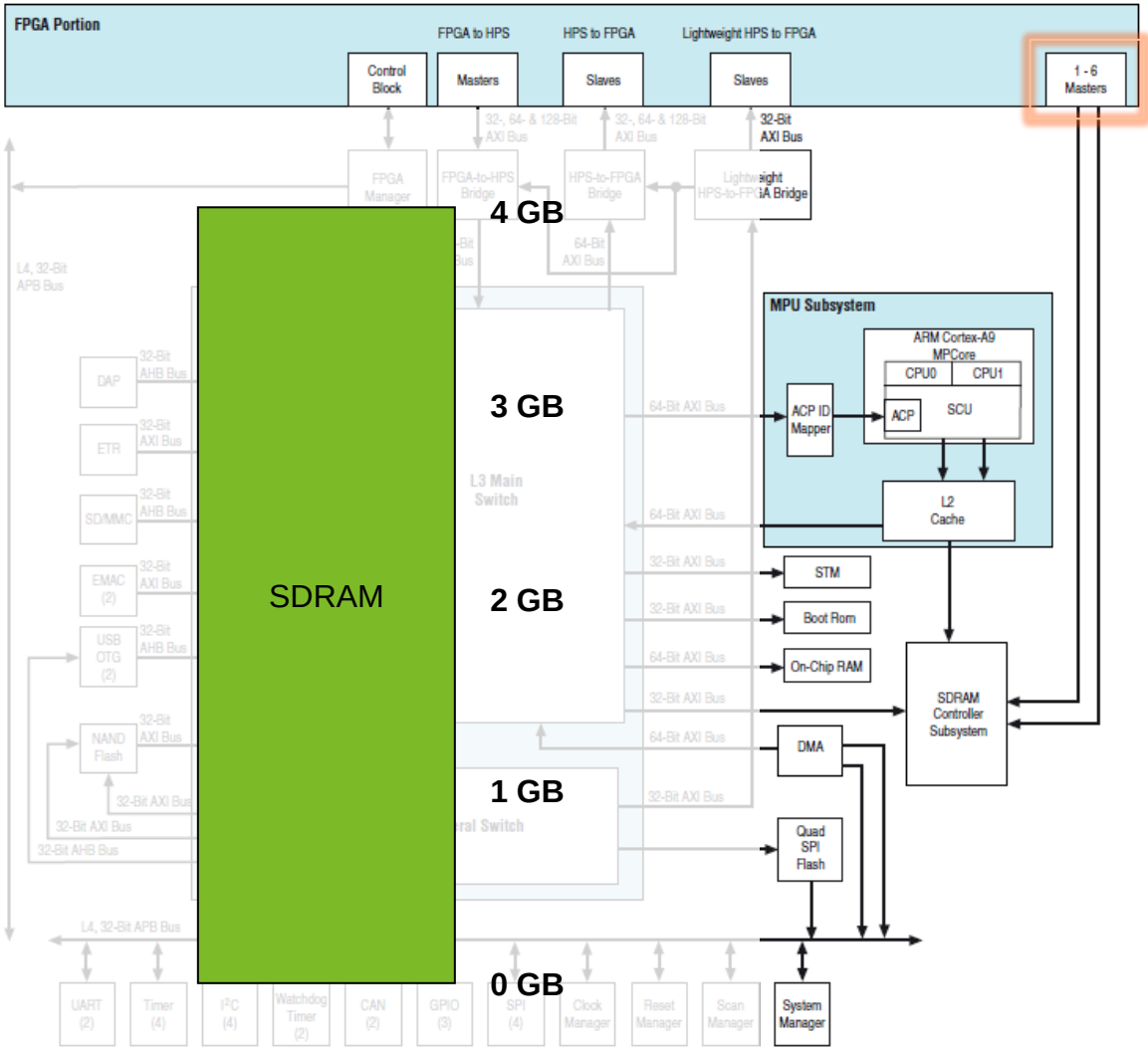


Arria 10 SoC HPS Physical Memory Map



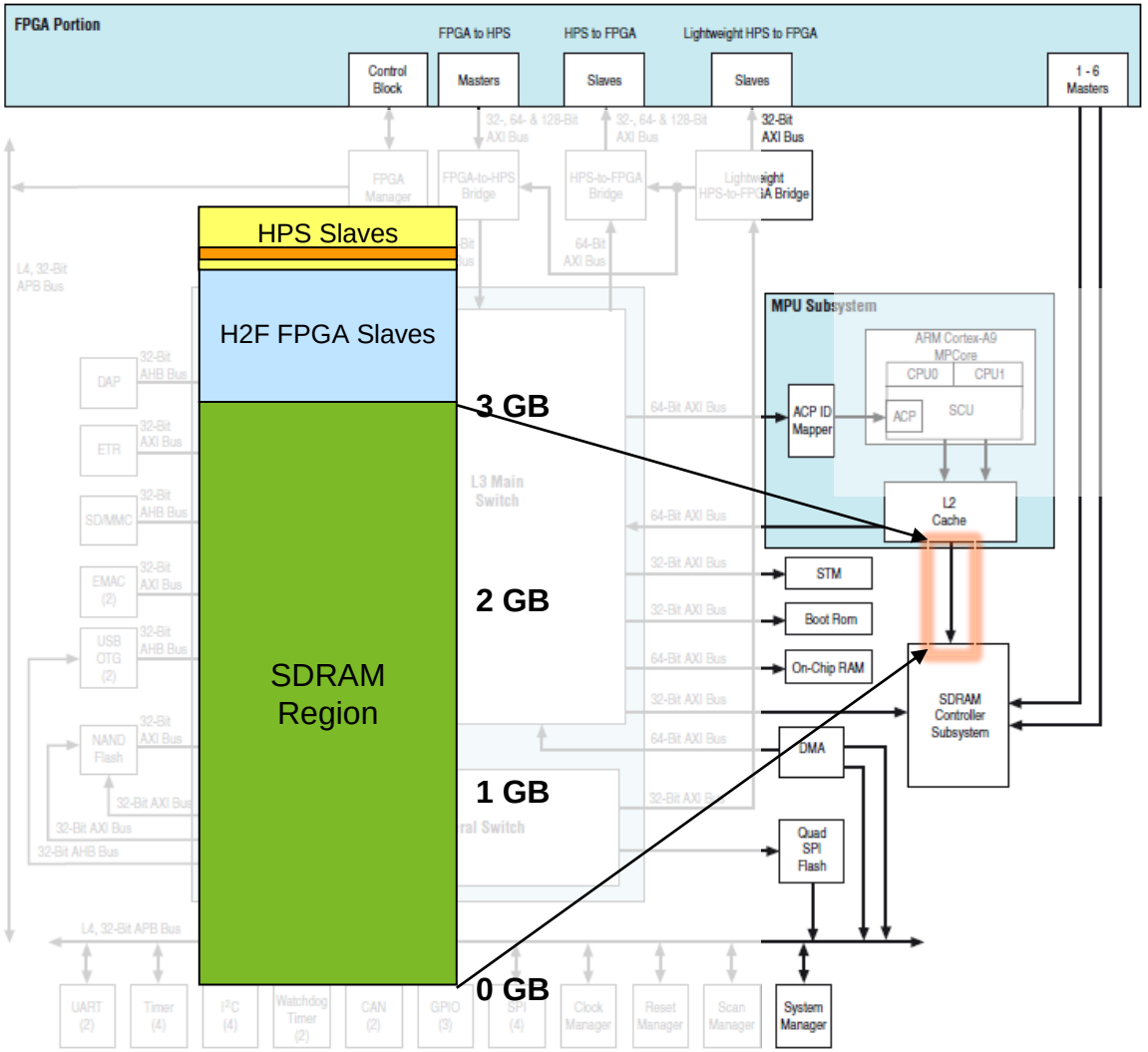
Physical Address Mapping

- FPGA Masters have access to full 4GB of SDRAM address space
  - Subject to MPFE MPU rules
  - No coherency
  - No virtual addressing



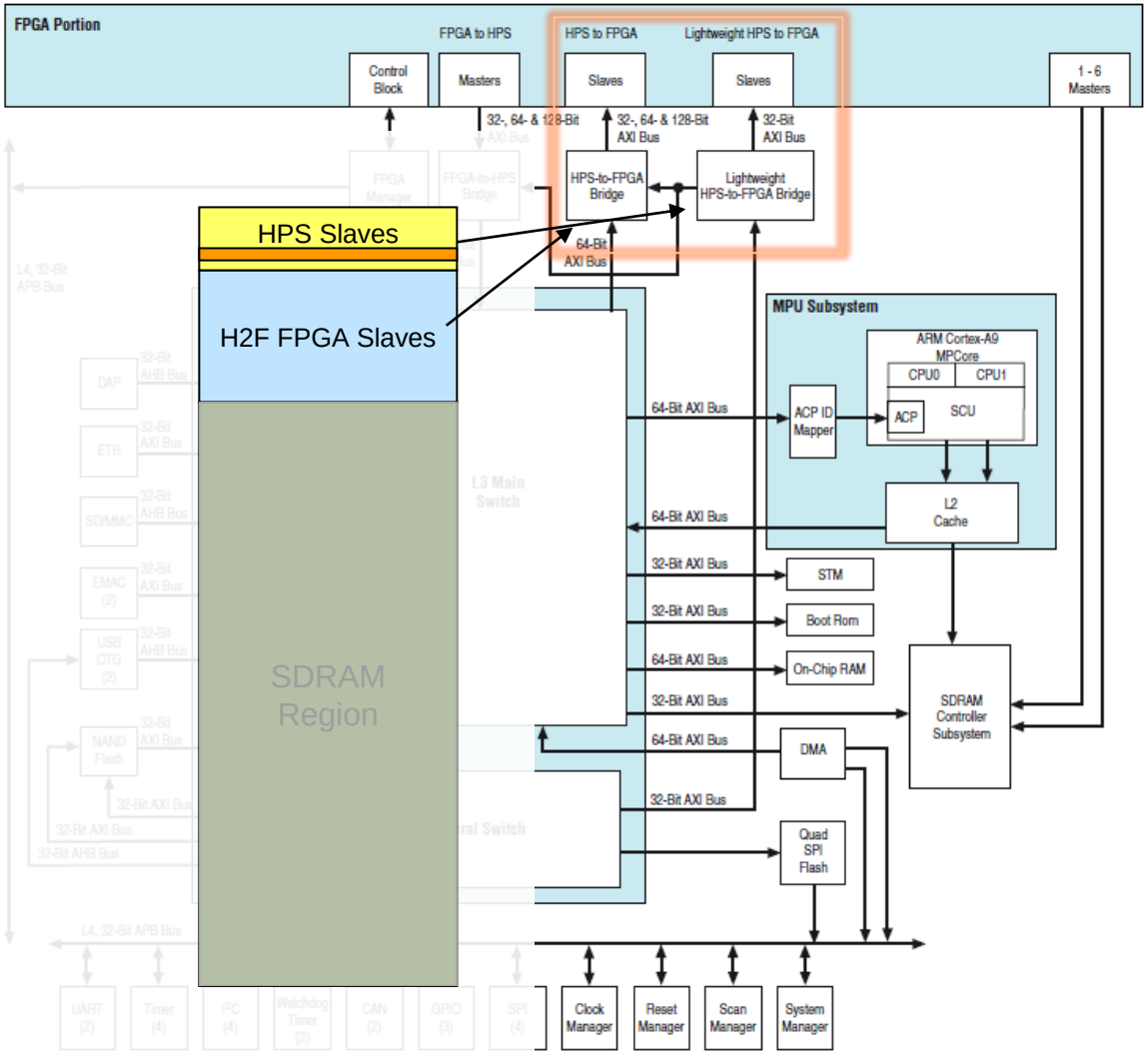
Physical Address Mapping

- MPU can only access the lower 3GB of SDRAM



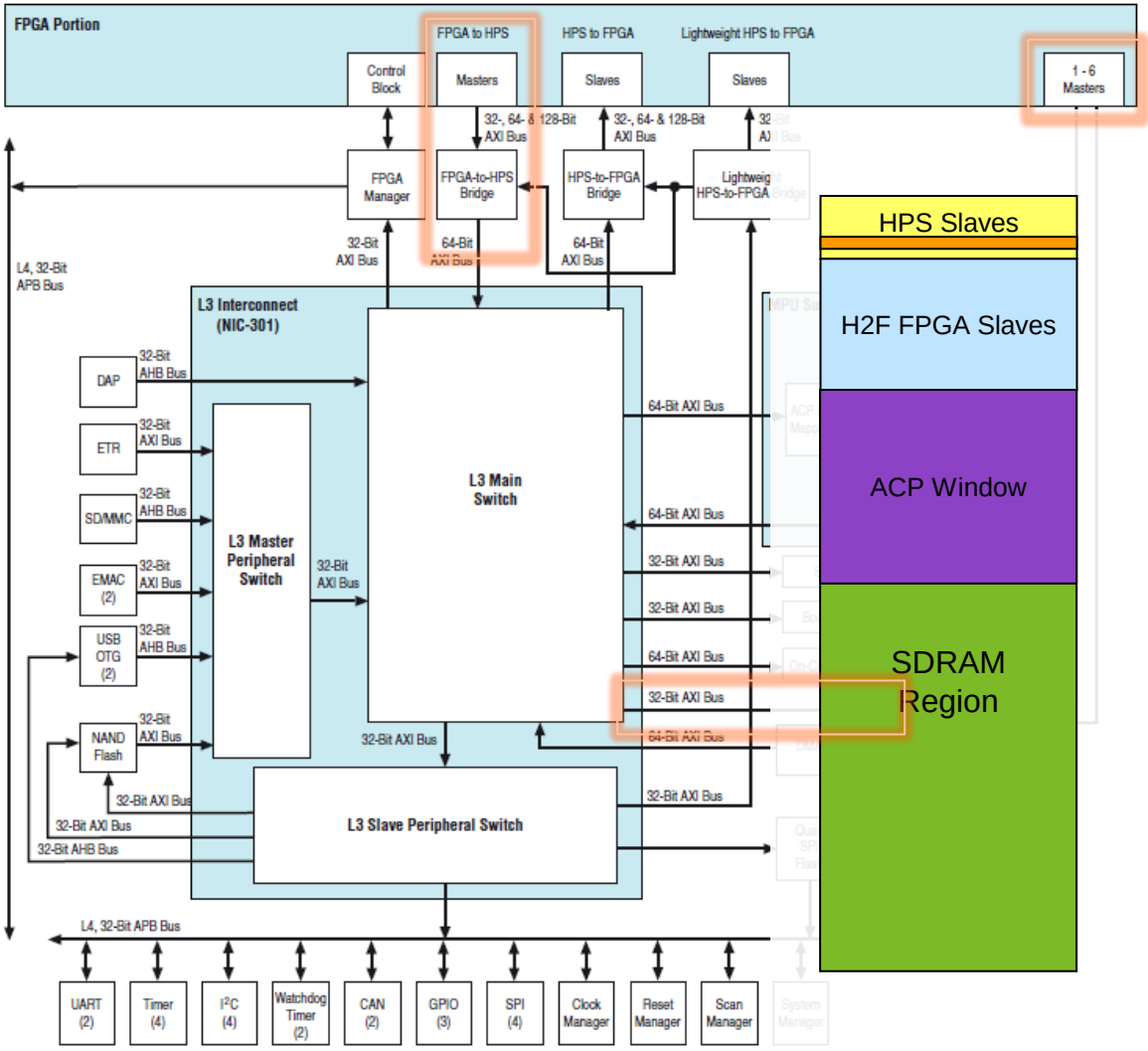
Physical Address Mapping

- MPU can access 960MB of FPGA address space via HPS to FPGA Bridge
- MPU can access 2MB of FPGA address space via HPS to FPGA Lightweight bridge
- See Developing Drivers for Altera SoC Linux Workshop

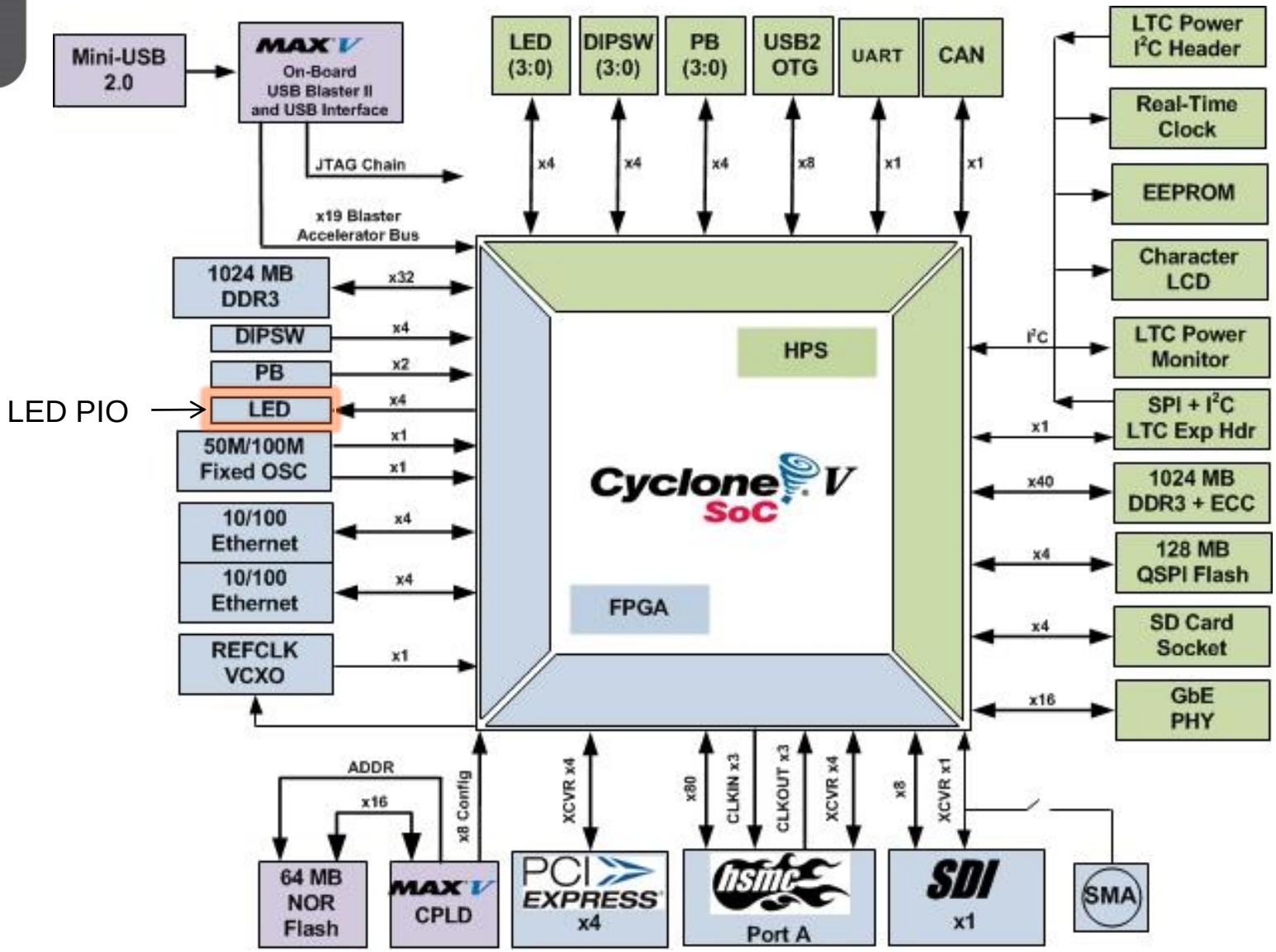


Physical Address Mapping

- FPGA to HPS (F2H) masters see 4 GB address space
  - 2GB SDRAM
  - 1GB ACP Window
  - 960MB H2F
  - 64MB HPS I/O
- F2H bandwidth to SDRAM limited vs. FPGA to SDRAM bridge

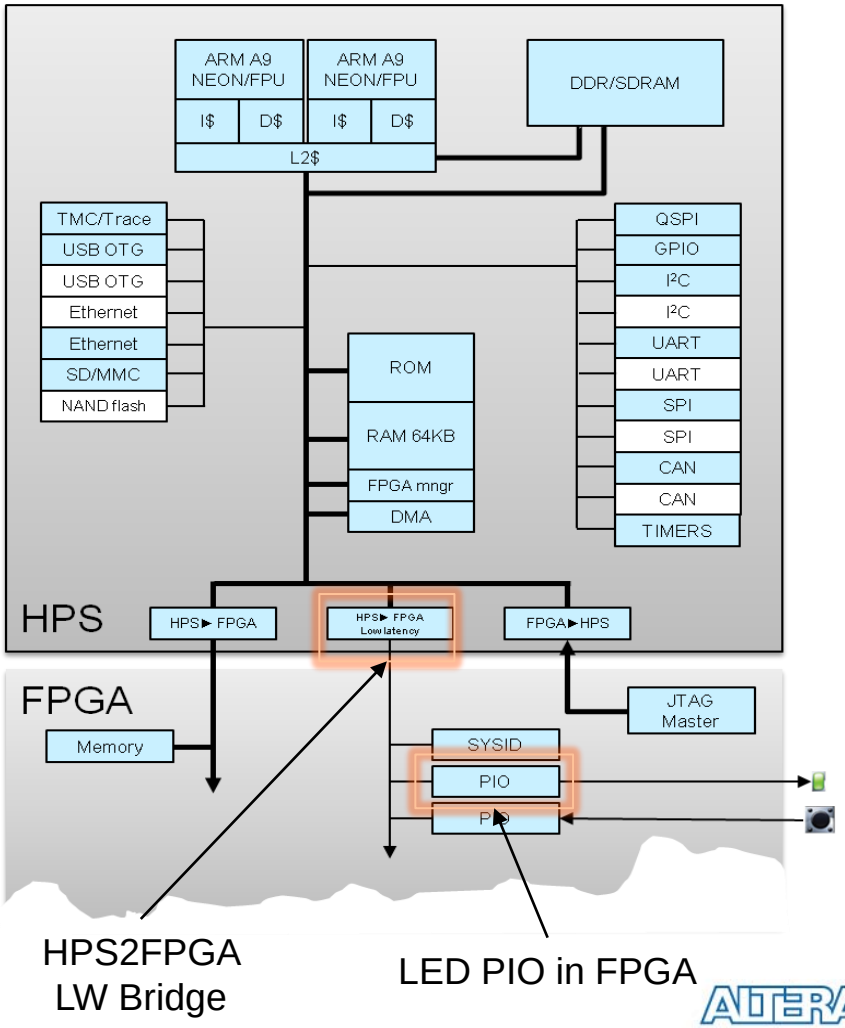


Cyclone V SoC Dev. Kit Memory Map Example

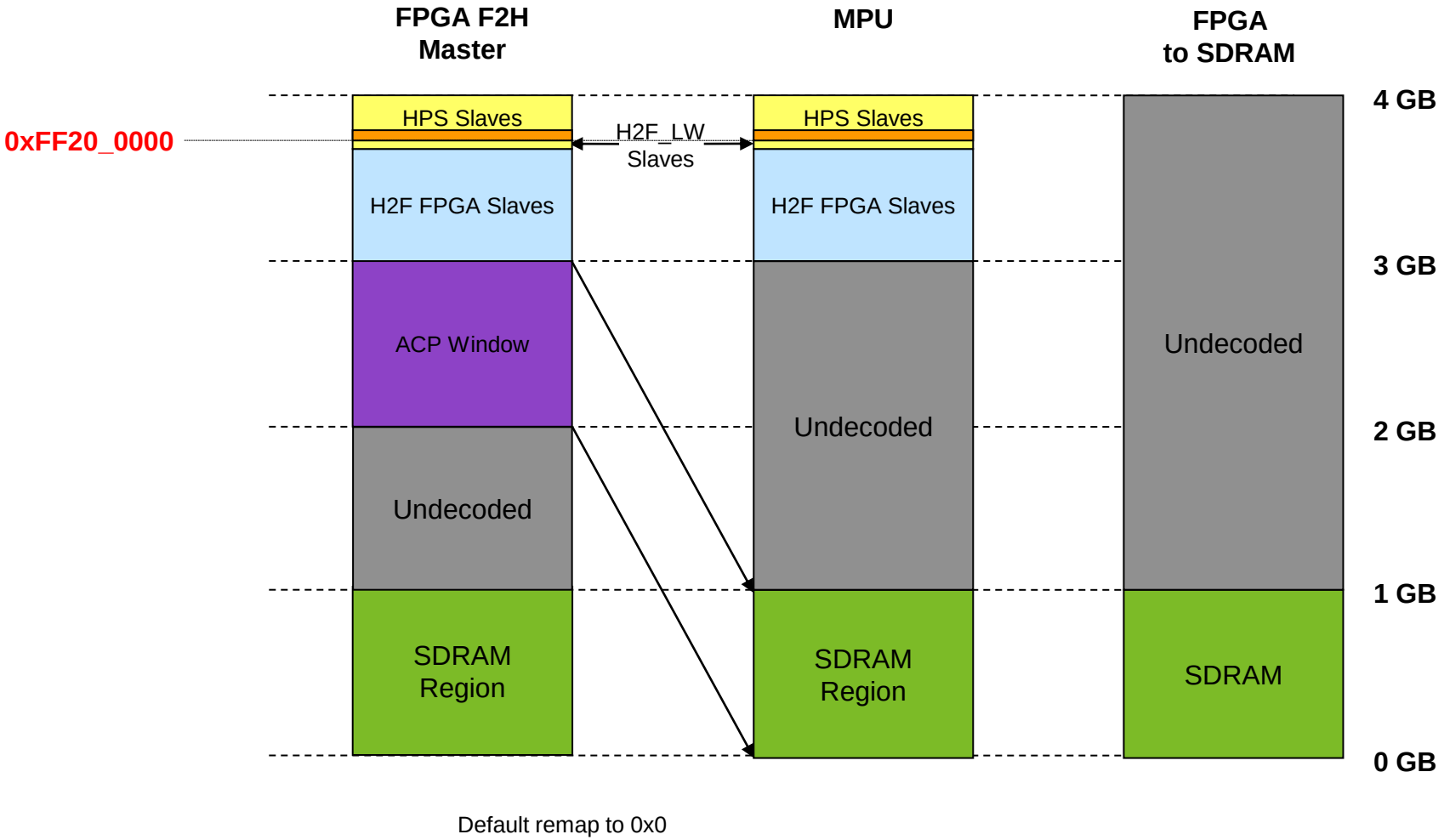


Golden Hardware Reference Design (GHRD)

- Complete hardware design
  - Most BSP use this design
  - Simple custom logic design in FPGA
  - All source code and Quartus II / Qsys design files for reference
  - Includes JTAG Master(s) for System Console access.
  - Similar for each Board



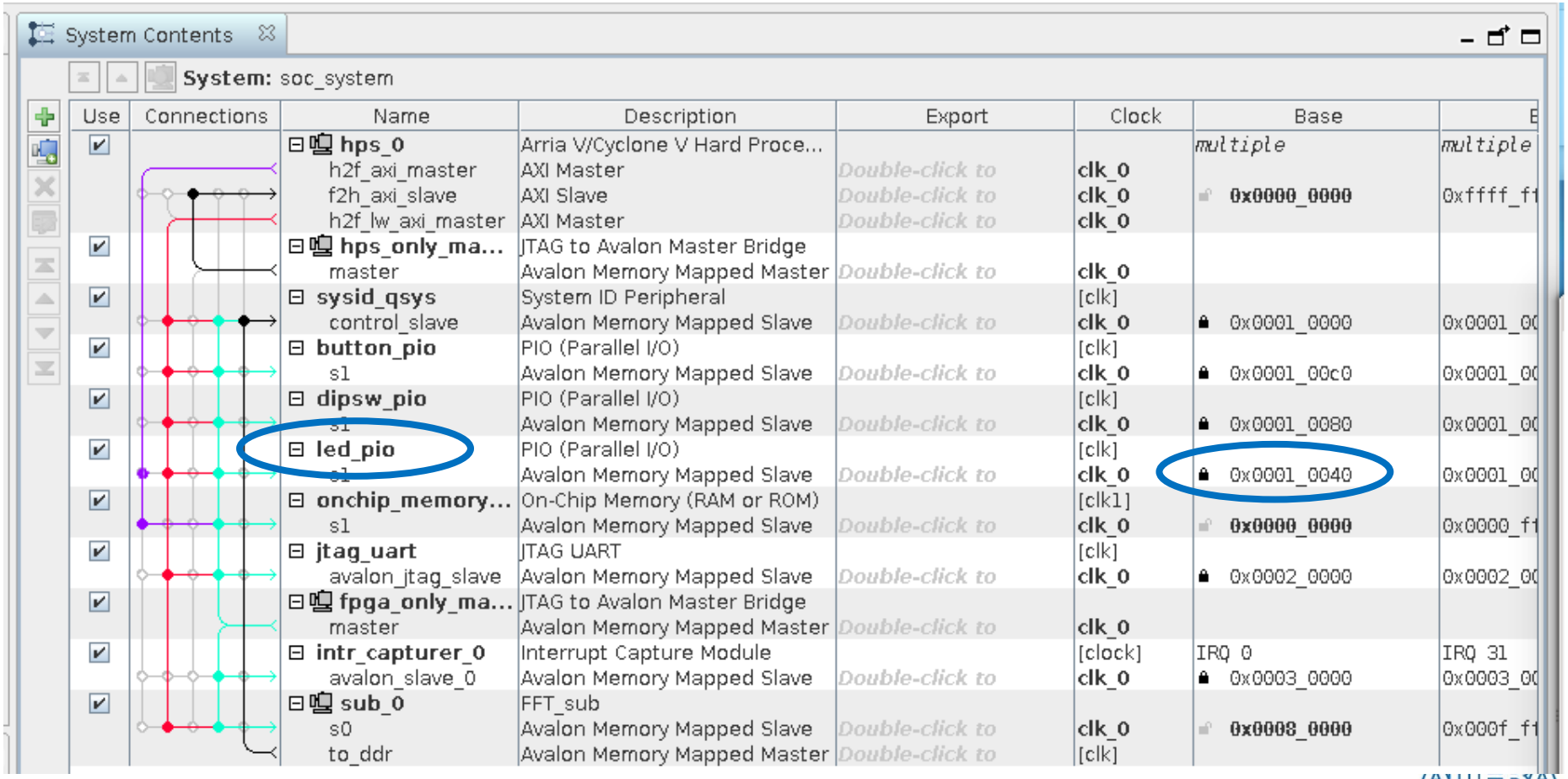
Cyclone V SoC Dev. Kit Memory Map Example



Default remap to 0x0

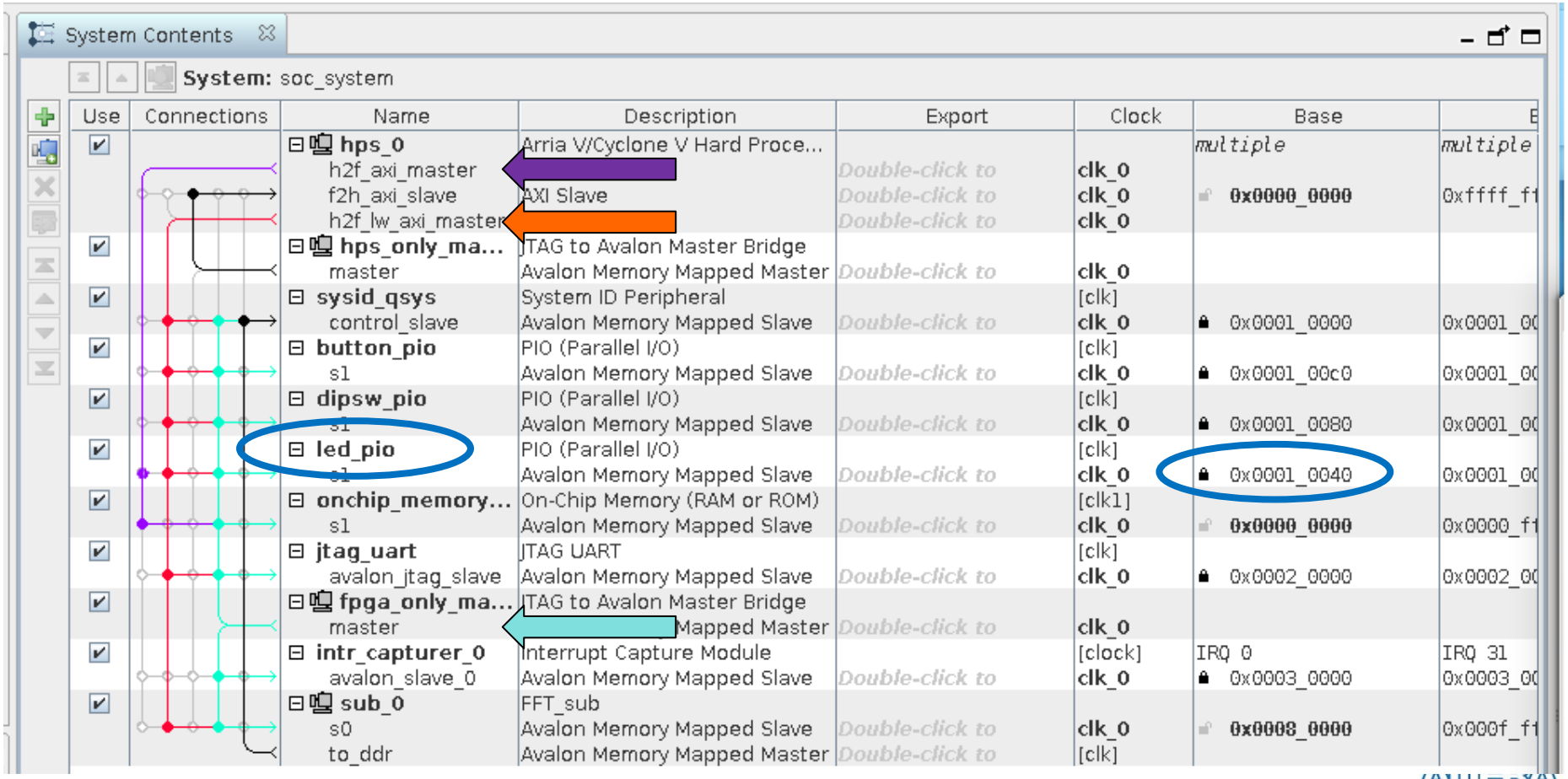
Memory Map through Qsys

- led_pio is at Address 0x0001_0040 in Qsys



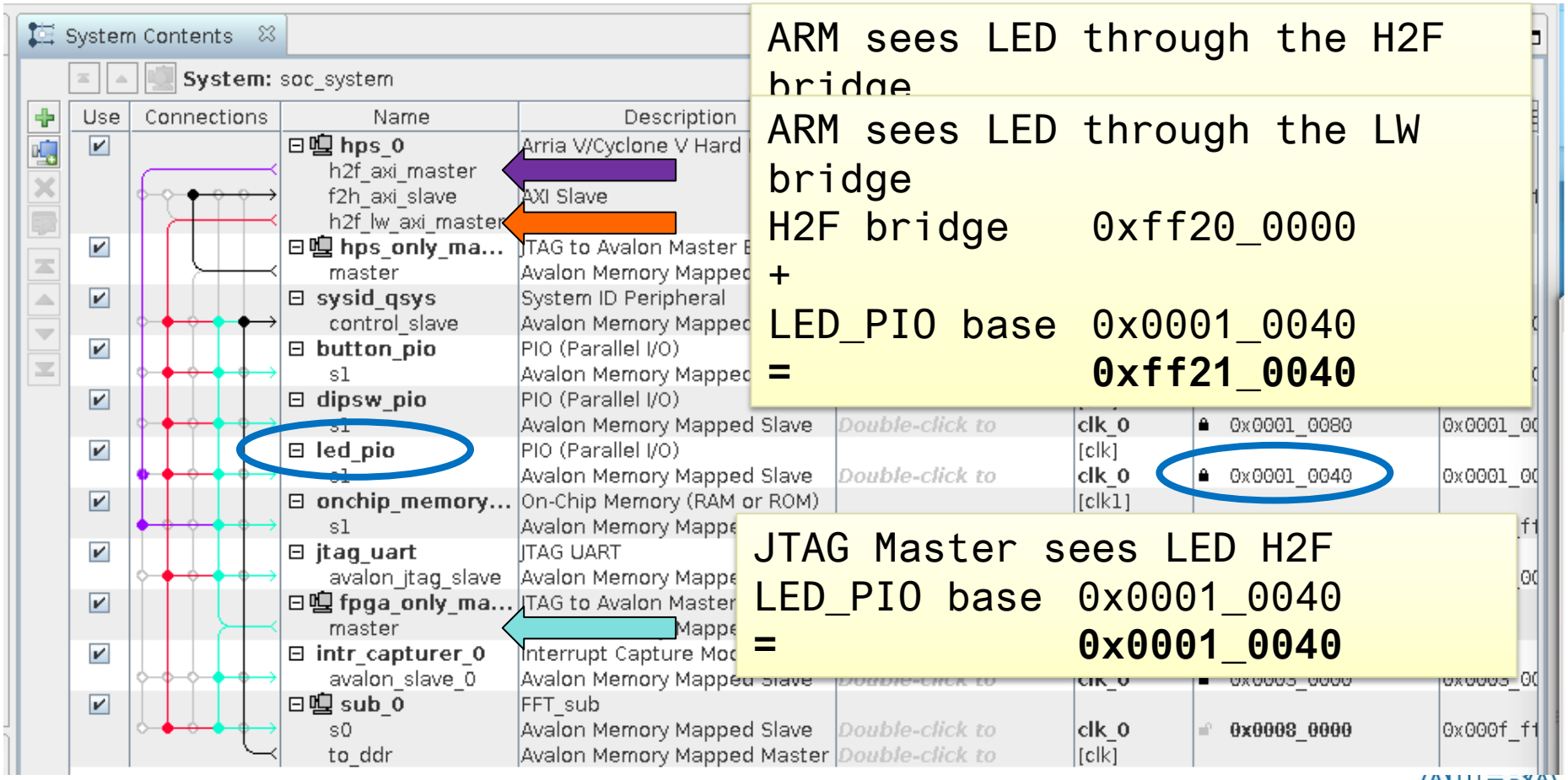
Memory Map through Qsys

- It is connected to 3 Masters



Memory Map through Qsys

- Each master sees the slave at a different address
  - HPS master addresses offset from base location of bridge slave in HPS



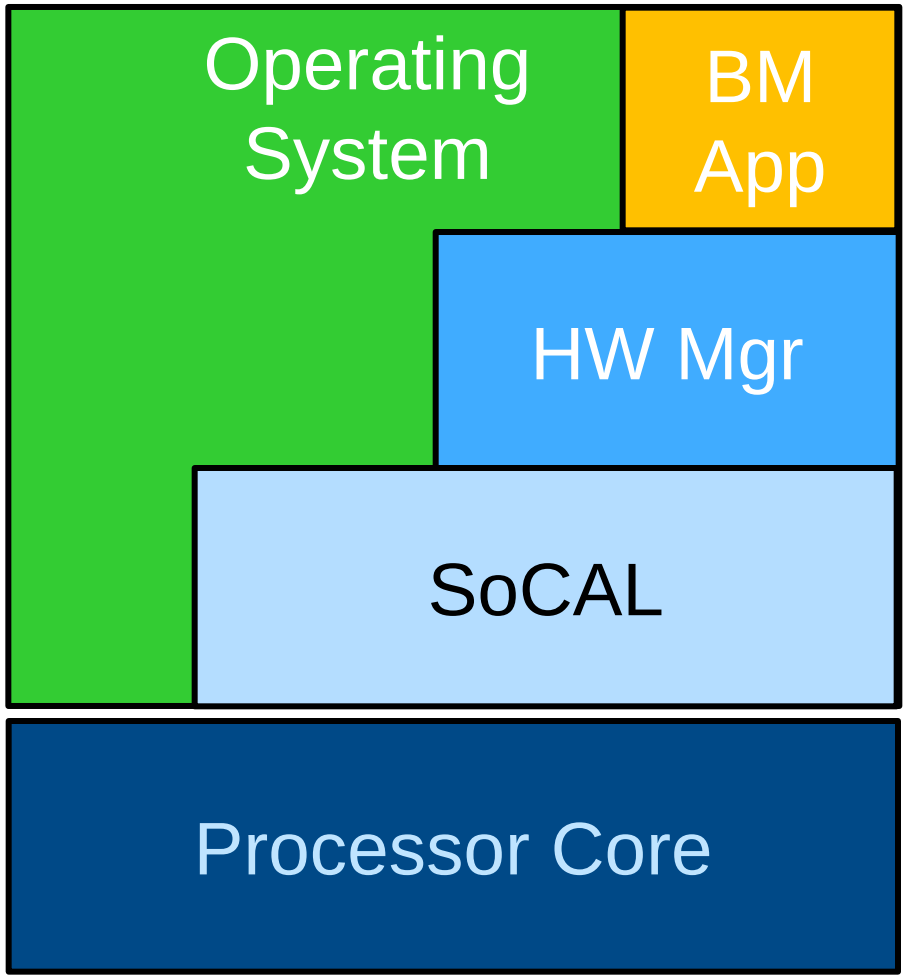
Altera SoC
Bare-Metal Software Development

Software For Bare-Metal Programming

- Hardware Libs (HWLIBs)
  - SoC Abstraction Layer (SoCAL) – Low-Level HAL
    - Header files
    - Documentation
  - Hardware Manager (HWMgr)
    - Adds C and some assembly language
    - #includes SoCAL header files
- SoC Embedded Development Suite (EDS)
  - IDE for development/debug of bare-metal application
  - Includes tools for flash programming, boot-image generation, and configuring and generating Preloader (Initial Program Loader)
- Bare-Metal Compilers
  - Altera GCC EABI Compiler
  - ARM Pro compiler (also included as of SoC EDS 14.0.2)

Hardware Libs Usage

- NO O/S (or BM App)
  - Simple executive
  - Polling Loop
  - Can use Bare-Metal tools
    - SoCAL, HW Mgr, Compiler
- Low-Level O/S
  - No device driver model
  - i.e. ThreadX, uC/OS
  - Can use some Bare-Metal
    - SoCAL, HW Mgr
- High-Level O/S
  - Device Driver Model
  - e.g. Linux, VxWorks
  - Can use some BM tools:
    - SoCAL, HW Mgr



SoCAL Overview

- Logical interface abstraction to a physical device or registers
- Low-level HAL
  - API closest to actual hardware
- Decouples software from hardware
  - Isolates client software from hardware changes
- Designed to be used with or without an OS
- Generated code
  - Preprocessor macros, enum, and struct declarations
  - Commented inline
- Targets C and assembly language programmers
- Defines programmatic access to hardware:
  - HPS address space
  - Hard IP components
  - Peripheral registers
  - Register fields
- Tested with GCC and ARM Pro bare-metal compiler tool-sets

Hardware Manager (HWMgr) Overview

- Functional APIs that implement more complex configuration and operational control over SoC hardware resources
  - Satisfy specific timing constraints
  - Error checking through parameter constraints and validation checks
  - Provides a level of precondition assertion checking
    - For example, checking that the FPGA is powered on and in USER mode prior to initializing an FPGA bridge
  - Provides a level of post condition assurance
    - For example, if the configuration of a scan chain fails
- Designed to be used with or without an OS
  - Open source BSD license (non GPL)
  - No namespace clash
  - Thread-safe

APIs in HWLIBs HWMgr

MPU Subsystem
- Memory Map Cntl
- Address Filters
- Mem Coherence

Timers
- Watchdog
- General Purpose

Cache/MMU
- Cache Mgmt
- MMU Mgmt

Serial
- UART
- SPI
- I2C
- CAN

FPGA Manager
- Full Configuration

Bridge Management
- FPGA2HPS
- HPS2FPGA
- LWHPS2FPGA

Flash Memory
- QSPI
- NAND
- SD/MMC

- Minimal Preloader
- Interrupt Ctrl
- Pin I/O Cnf Mgmt
- ECC Mgmt
- Parity Mgmt

- Clock Manager
- Reset Manager
- System Manager
- SDRAM Ctrl
- GPIO
- DMA

SoCAL Layer (non ARM IP)

Current **Future rel.**

Software Delivery

- HWLIBs and Bare-Metal application development components are included with SoC Embedded Development Suite (SoC EDS)
- SoC EDS is distributed through the Altera Download Center
- Updates for HWLIBs are also be available between SoC EDS releases
  - See Altera.com -> Download Center -> “SoC RTOS and HWLIBs Support”

HWLIB Examples

- Hello World
- UART/Interrupt processing
- FPGA Programming
- Error Correction Code (ECC)
- Web Server/Ethernet LWIP (through codetime.com)
- Minimal Preloader (MPL) – Non-GPL Preloader
- Many more examples available to download
  - https://www.altera.com/support/support-resources/design-examples.html#soc-design-examples

HWLIB Examples – Minimal Preloader (MPL)

- Non-GPL bootloader for Cyclone V and Arria V
  - Replaces default GPL Preloader
  - UEFI-based preloader will be available for Arria 10
- Runs in Preloader boot stage
  - First programmable image loaded by bootrom
- Loads the next stage bootloader or executable image
  - Typically loads a bootloader or IPL (Initial Program Loader)
  - Can load image from QSPI Flash or SD/MMC Flash
  - Boot from FPGA option coming soon.
- Configures clocks, PLLs, memory controller, IOCSR
- Distributed as HWLIBs example starting in SoC EDS 14.0.1

Learn More

- SoC EDS user guide
  - https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_soc_eds.pdf
- Bare metal compiler

The bare-metal compiler comes with full documentation, located at:

<SoC EDS installation directory>/host_tools/mentor/gnu/arm/baremetal/share/doc/sourceryg++-arm-altera-eabi

The documentation is offered in four different formats to accommodate various user preferences: Html files, Info files, Man pages, PDF files

- HWLIBs and SoCAL

Reference documentation for the SoCAL API and HW Manager API is distributed as part of the SoC EDS Toolkit. This reference documentation is provided as online HTML accessible from any web browser.

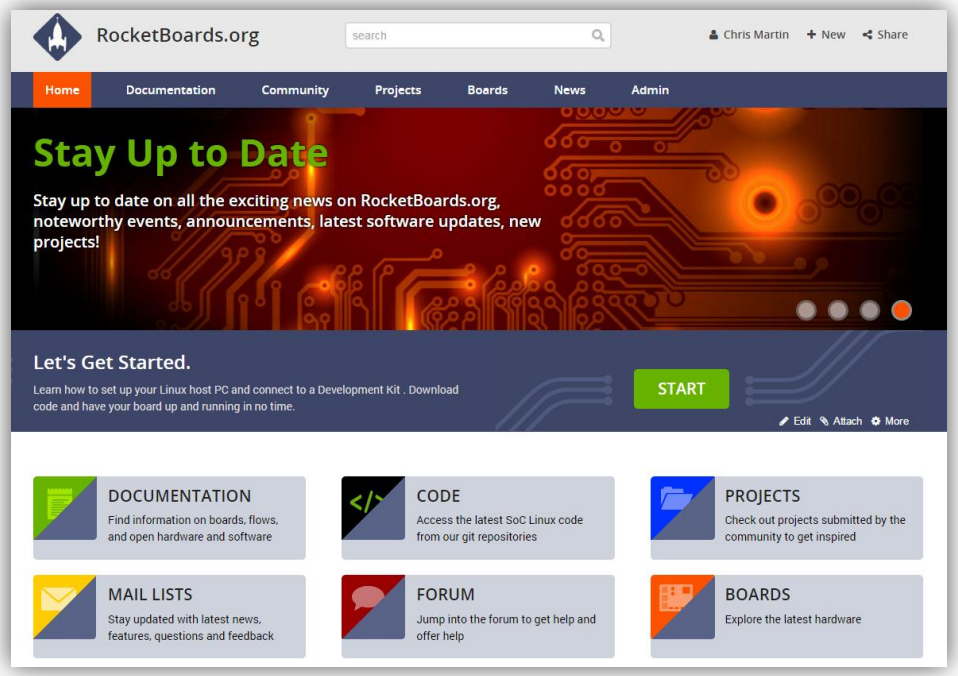
The locations of the online SoC FPGA Hardware Library (HWLIB) Reference Documentation are:

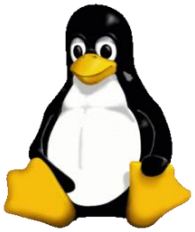
- SoC Abstraction Layer (SoCAL) API Reference Documentation:
  <SoC EDS installation directory>/ip/altera/hps/altera_hps/doc/socal/html/index.html
- Hardware Manager (HW Manager) API Reference Documentation:
  <SoC EDS installation directory>/ip/altera/hps/altera_hps/doc/hwmgr/html/index.html.

Altera SoC Linux Overview

Linux for Altera SoCs

- High Quality Linux Support
- Modern release strategy
- Multiple Kernel Versions
- Community Enablement





LONG TERM SUPPORT INITIATIVE

Ångström

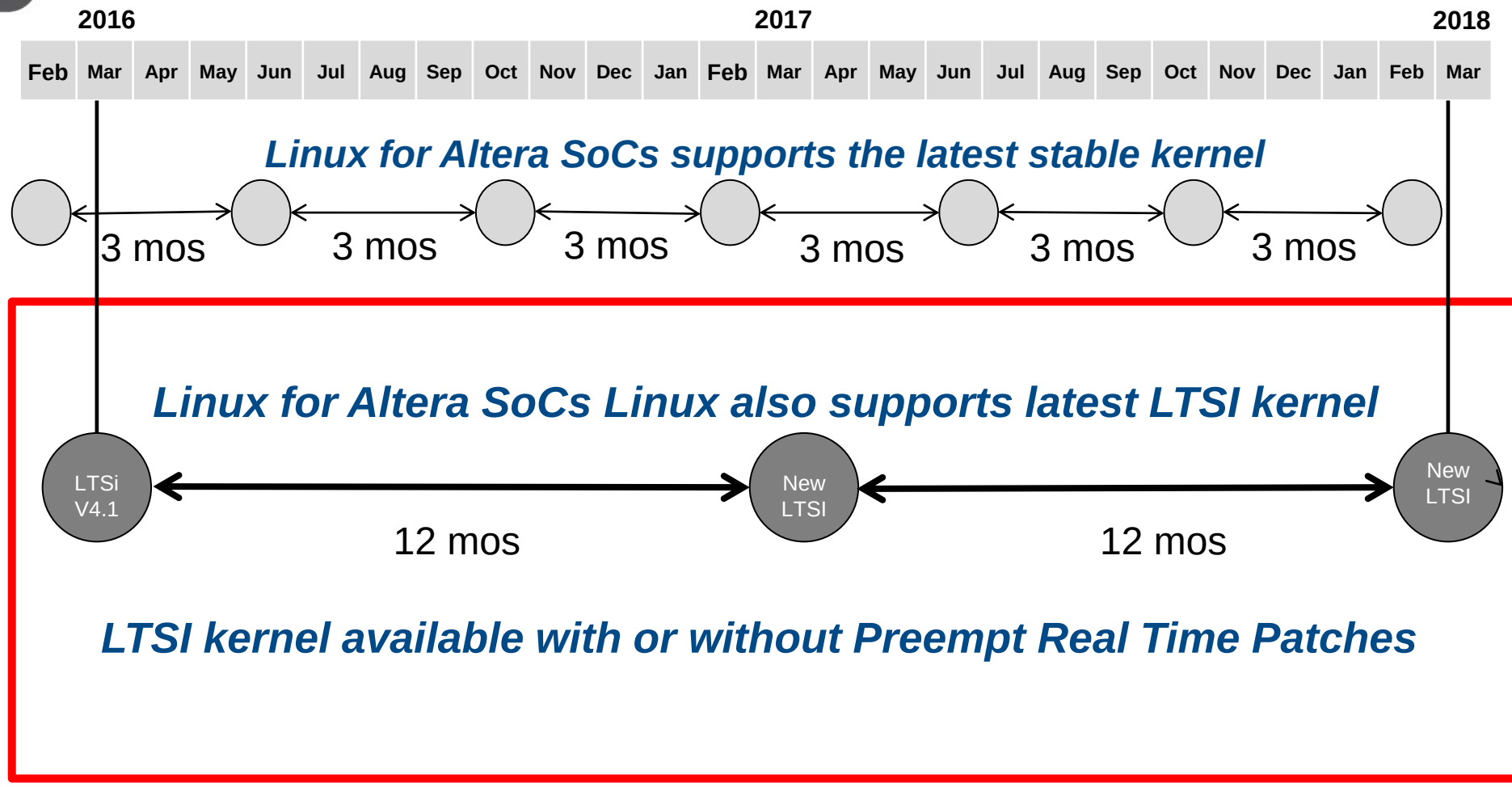
git

Linaro

yocto PROJECT

ALTERA now part of Intel

Altera SOC Linux Provides Customers Kernel Choices

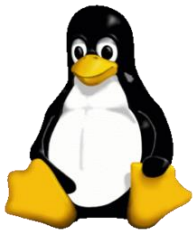


Industry-leading Linux support

- Altera keeps up with the Linux community
  - Kernel is upgraded every 3 months against every new kernel.org release
  - Altera is the maintainer for the SoCFPGA architecture folder (mach-socfpga)
- Altera maintains the LTSI kernel
  - Altera SOC Linux LTSI v3.10, more stable code base for 24 months
  - Fixes, improvements and new features back-ported from latest kernel
  - Seamless transition to commercial Linux vendors, Wind River, etc.
- Altera's SoC drivers have high quality
  - Altera SoC Hard IP acquired from EDA vendors, use community Linux drivers
  - Altera SoC Linux drivers used by many SOC vendors, therefore drivers have more support and higher quality
  - Altera upstreams fixes and improvements to code to the kernel and u-boot
- Altera supports a modern release strategy
  - Updates Public GIT trees every 2 weeks – complete transparency
  - Published on community site: See git.rocketboards.org
  - See the NEWS section of RocketBoards.org latest updates

Altera SoC Linux Support Model

- Rocketboards.org
  - SoC & Nios II Linux documentation
  - SoC & Nios II SoC Linux reference & example designs
- Rocketboards.org RFI & Linux Community
  - Kernel/RFS/u-boot questions
  - SoC/Nios II subsystem and driver questions
- Altera.com and Rocketboards.org
  - SoC EDS & Quartus/QSys documentation and questions
  - SoC Preloader questions
  - SoC HPS implementation specific questions
  - Use myAltera for service requests
- Support from Altera is focused on SoCFPGA and NiosII Linux Board Support Package
- Altera enables Linux development on SoCFPGA & Nios II



Linux Resources

now part of Intel

Linux Documentation Resources

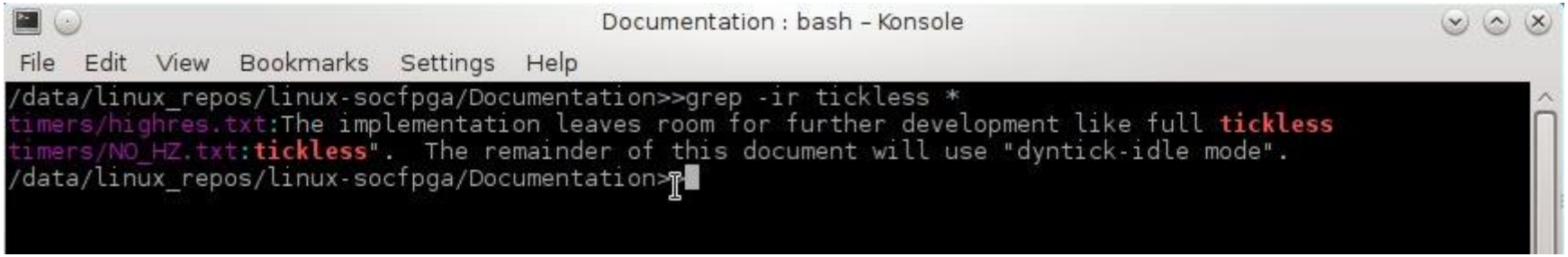
- GIT
  - Distributed revision control system to enable distributed collaboration
  - On-line documentation & training:
    - http://git-scm.com/doc
    - https://training.github.com
- Denx u-boot Manual
  - Complete documentation from the folks who wrote Das u-boot
    - http://www.denx.de/wiki/U-Boot/Documentation
- Free-Electrons:
  - Complete training materials posted free
    - http://free-electrons.com/docs/
- Device Tree for Dummies
  - http://events.linuxfoundation.org/sites/events/files/slides/petazzoni-device-tree-dummies.pdf

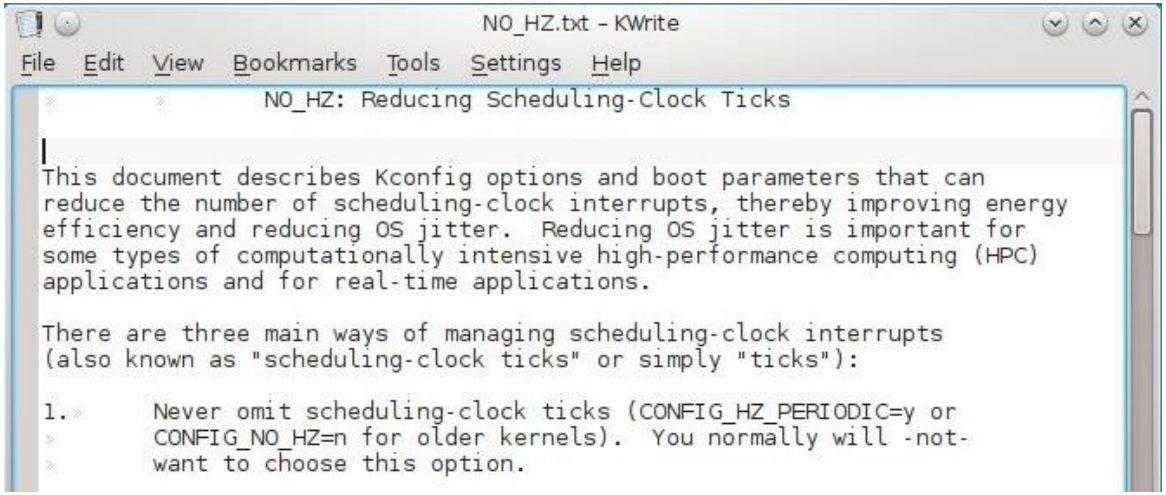
Linux Documentation Resources

- Yocto Project
  - https://www.yoctoproject.org/documentation
- Angstrom Distribution
  - http://www.angstrom-distribution.org/
- Open Embedded
  - http://www.openembedded.org/wiki/Main_Page

The Two Best Sources for Linux Development Information

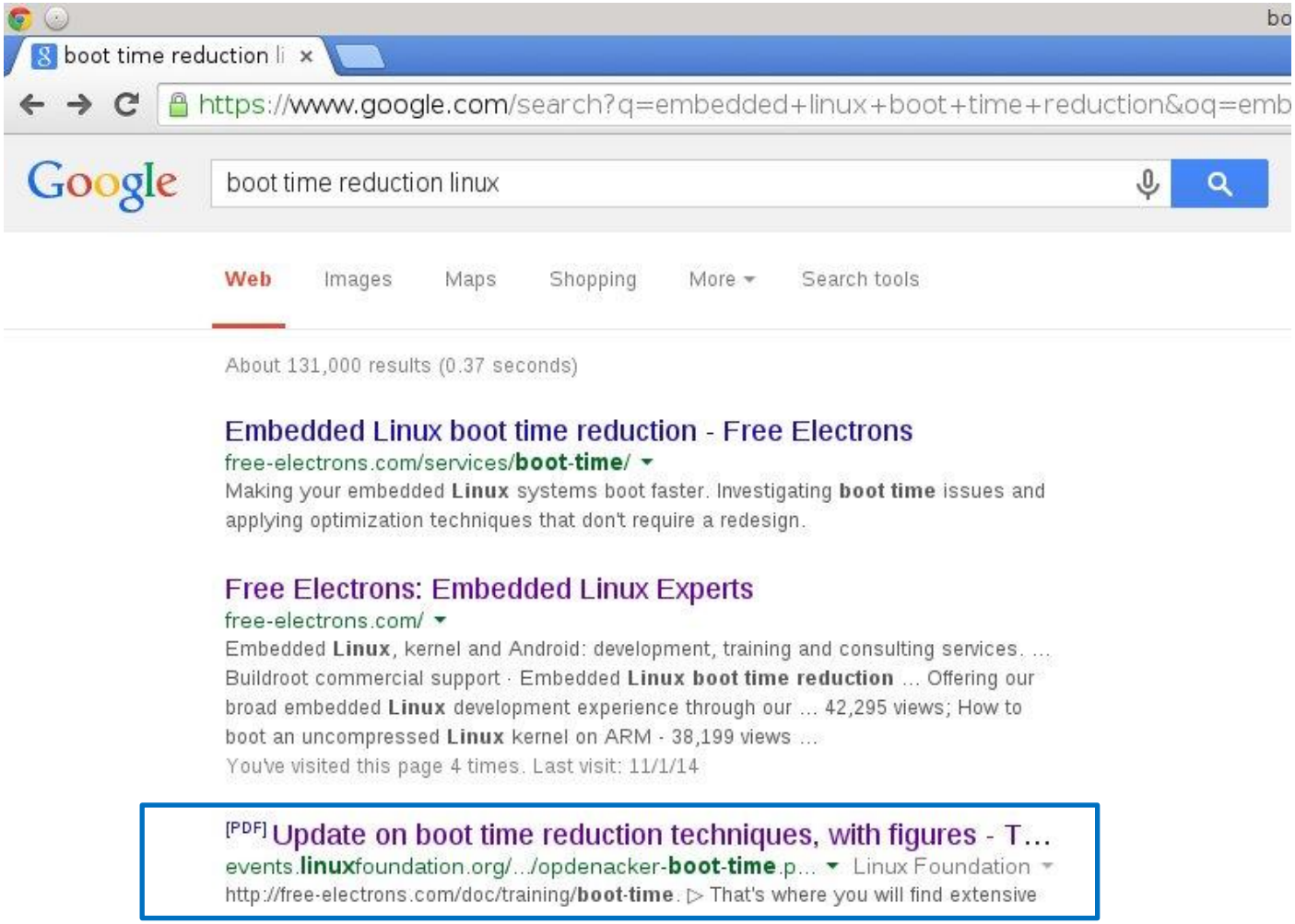
- Linux Kernel Documentation
  - The most complete and most essential Linux kernel documentation
  - Included with the Linux kernel source code
    - <local GIT repo>/Documentation





The Two Best Sources for Linux Development Information

- An open source OS breeds open source information



RocketBoards.org – Altera SoC Linux Community Portal

- The source for SoC FPGA Linux info
  - Golden System Reference Design (GSRD)
  - Updates on latest releases
  - Step-by-step getting started guides
- SoC FPGA Mailing List - RFI
  - Active community participation in answering SoC FPGA and Linux questions
- Example Projects, Applications, and Designs
  - From Altera and the SoC community
- Enables the SoC community to support Linux

RocketBoards.org

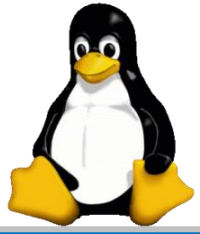
Learn More about SoCFPGA Linux

- SW Workshop #1 – Altera SoC SW Development Overview
- **SW Workshop #2 – Introduction to Linux on Altera SoC**
- **SW Workshop #3 – Developing Drivers for Altera SoC Linux**

Altera SoC
Operating System Support

Embedded OS Availability

ENEA Micriμm QNX WIND RIVER



Vendor	OS/RTOS	Development Tools	Available From
Open Source	Linux (current and 3.10 LTSI)	Linaro compiler	rocketboards.org
Wind River Systems	VxWorks 6.9.3 and 7.0	Wind River Workbench	Wind River
Micriμm	μC/OS-II, μC/OS-III	GNU compiler	Micriμm
Enea	OSE 5.5.3	Optima 2.6	ENEA
Express Logic	ThreadX G5.5.5.0	GNU compiler	Express Logic
Wind River Systems	Wind River Linux 5 and 7	Workbench/GNU	Wind River
QNX	QNX/Neutrino 6.5.3 and 6.6	Momentics	QNX
Fujisoft	Android	GNU compiler	Fujisoft
Green Hills	INTEGRITY	Multi/Green Hills	Green Hills
DDC-I	Deos	DDC-I	DDC-I
Code Time	Multicore Abassi	ARMCC/GCC	Code Time
Mentor	Nucleus	GCC	Mentor
eCosCentric	ECOSPRO (eCos)	GCC	eCosCentric

DDC-I Safety Critical Software Solutions for Mission Critical Systems

eCosCentric

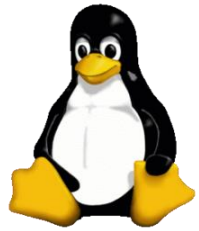
expresslogic

CODE TIME TECHNOLOGIES NEVER STOP INNOVATING

Green Hills SOFTWARE

ALTERA now part of Intel

Embedded OS Availability (page 2)



Vendor	OS/RTOS	Development Tools	Available From
MRA Digital	Android	GCC	MRA Digital
FreeRTOS	RTE	ARM DS-5 and GCC	Freertos.org
Monta Vista	CGE7 Linux	Monta Vista/GCC	Monta Vista
AUTOSAR	AUTOSAR 4.0.3 MCAL	Elektrobit Tresos Studio	Altera
Microsoft	Windows Embedded 7	Microsoft/Studio	Coming
Quadros	RTXC	GCC	Coming
rtems.org	RTEMS	GCC	Coming

Embedded SW Operating System Ecosystem (Japan)

Vendor	OS/RTOS	Development Tools	Availability
eSOL	eT-Kernel (uITRON4.0)	eBinder	eSOL
eForce	uC3 (uITRON 4.0)	ARMCC/SoCEDS	eForce
Toppers	Toppers (uITRON extended)	EDS/Toppers	Toppers
Mispro	NORTi(uITRON 4.0 standard)	ARMCC	Mispro

Broad JTAG Debugging Tools Support

Company	Debugger
Altera	USB Blaster II
Lauterbach	Trace32
ARM	DSTREAM
Wind River	ICE II, Probe
Green Hills	Probe
Yokogawa Digital Computer	AdviceLUNA
Kyoto Microcomputer	Partner-Jet
Computex	PALMiCE3
Segger	J-Link
iSystem	Coming Soon
Ronetix	PEEDI

OS support

- List
  - https://www.altera.com/products/soc/ecosystem.html

Preloader Generation

For Cyclone V and Arria V SoC

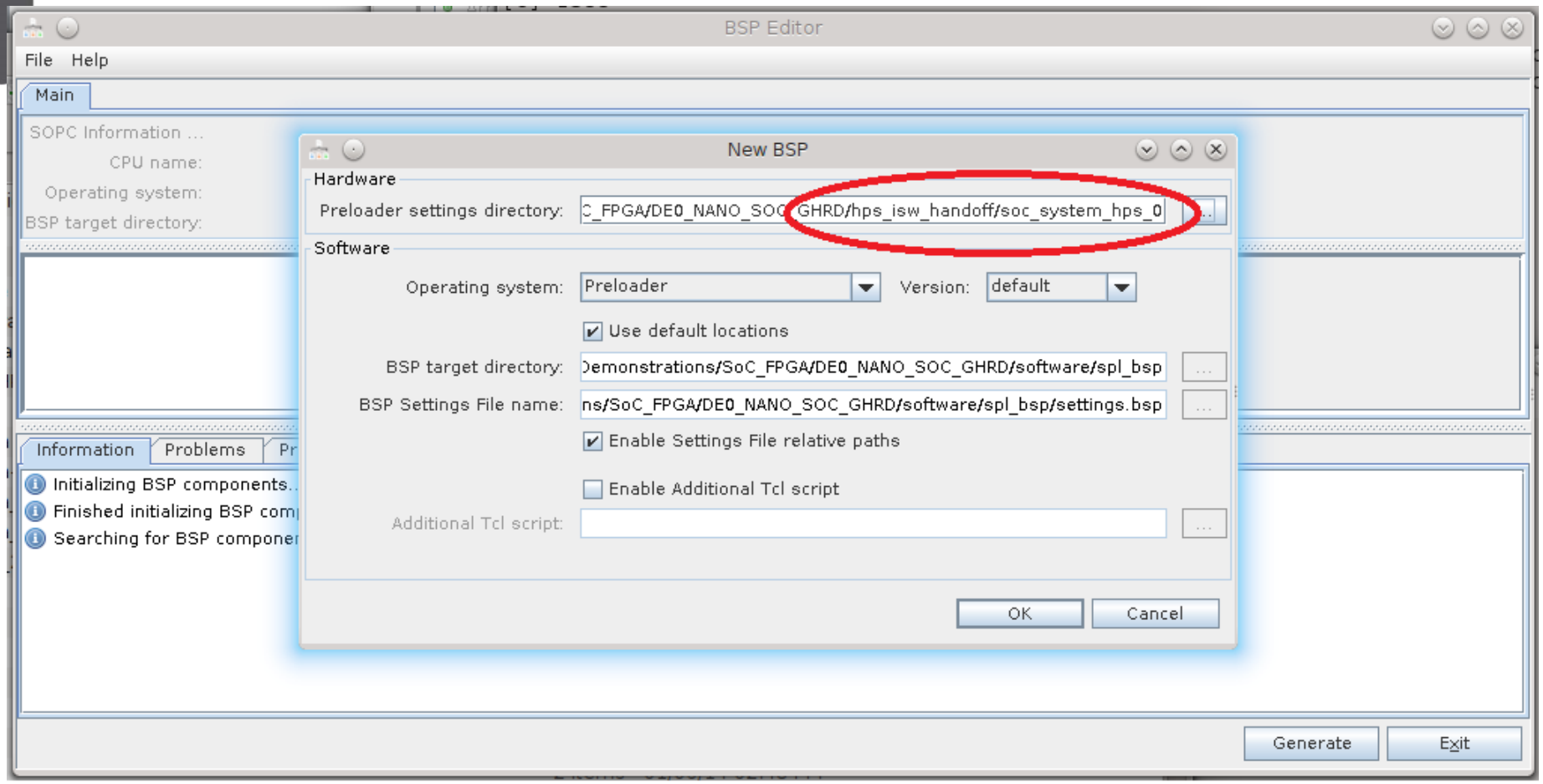
Preloader

- Runs after the boot ROM
- Configures the HPS IO
- Configures and Calibrates the DDR controller
- Optionally loads the FPGA image
  - http://www.rocketboards.org/foswiki/Documentation/GSRD131ProgrammingFPGA
- Is just a small version of u-boot called u-boot-spl

Generating the preloader

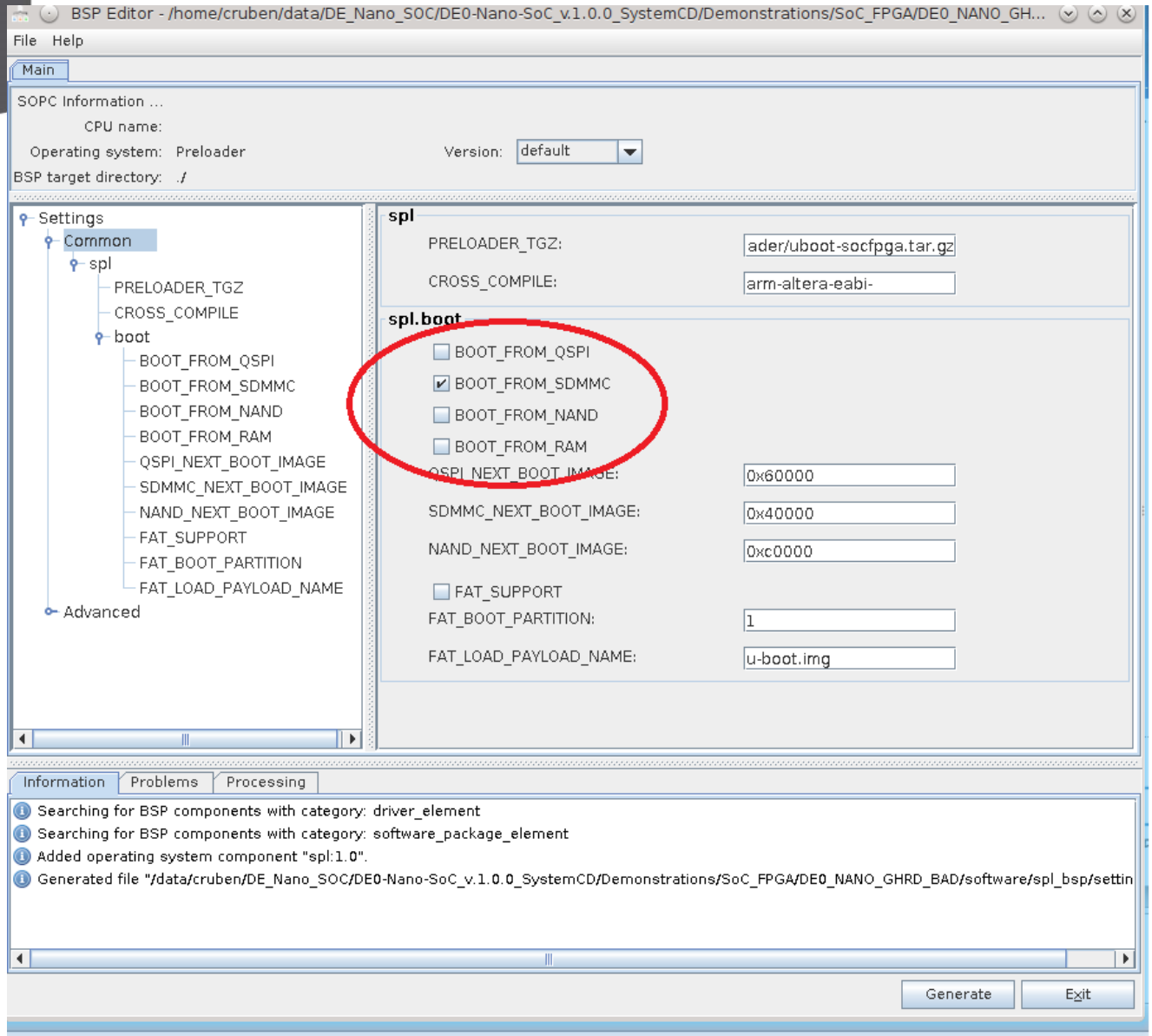
- GUI and Command line options
  - http://www.rocketboards.org/foswiki/Documentation/GSRD131Preloader
- Launch in embedded shell
  - “bsp-editor &”
- Choose File -> New
- Select the folder under the handoff folder

bsp-editor - Choose Soc System



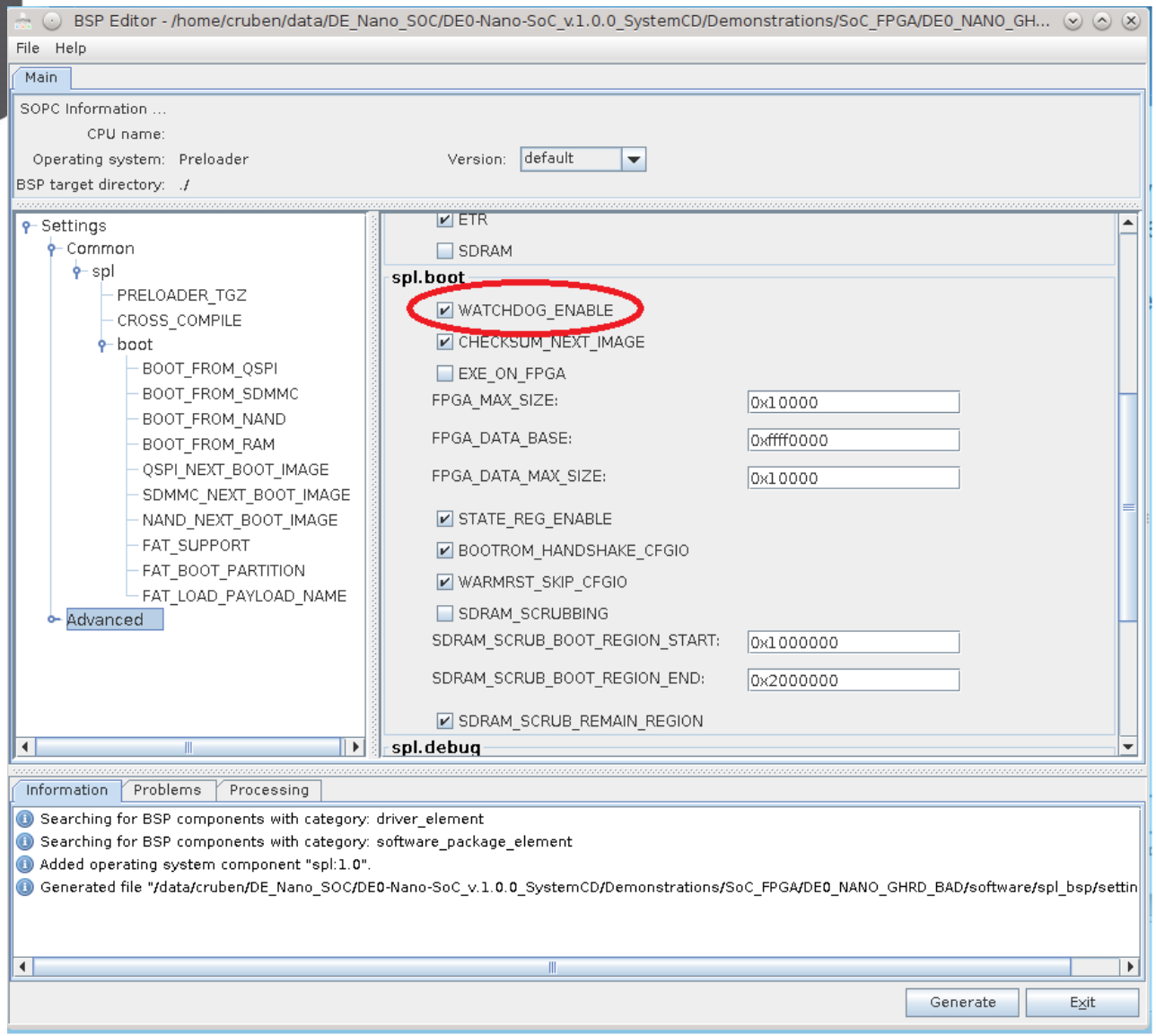
- Select OK

Bsp-editor - Choose Next Boot location



Choose the Device that contains the next executable.
Note: Only select one

Preloader - Advanced Settings



Choose the advanced settings you may want.

i.e. you may not want watchdog for bare metal system.

Select Generate
Then at the embedded prompt cd to the BSP directory and type "make"

Build Preloader from the command line

```
bsp-create-settings \
                --bsp-dir "./qspi_preloader" \
                --preloader-settings-dir "$(SOCEDS_ROOT)/examples/hardware/cv_soc_devkit_ghrd/hps_isw_handoff/soc_system_hps_0" \
                --settings "./qspi_preloader/settings.bsp" \
                --type spl \
                --set spl.CROSS_COMPILE arm-altera-eabi- \
                --set spl.PRELOADER_TGZ "$(SOCEDS_ROOT)/host_tools/altera/preloader/uboot-socfpga.tar.gz" \
                --set spl.boot.BOOTROM_HANDSHAKE_CFGIO 1 \
                --set spl.boot.BOOT_FROM_QSPI 10\
                --set spl.boot.BOOT_FROM_RAM 0 \
                --set spl.boot.BOOT_FROM_SDMMC 1 \
                --set spl.boot.CHECKSUM_NEXT_IMAGE 1 \
                --set spl.boot.EXE_ON_FPGA 0 \
                --set spl.boot.FPGA_DATA_BASE 0xffff0000 \
                --set spl.boot.FPGA_DATA_MAX_SIZE 0x10000 \
                --set spl.boot.FPGA_MAX_SIZE 0x10000 \
                --set spl.boot.QSPI_NEXT_BOOT_IMAGE 0x60000 \
                --set spl.boot.SDMMC_NEXT_BOOT_IMAGE 0x40000 \
                --set spl.boot.STATE_REG_ENABLE 1 \
                --set spl.boot.WARMRST_SKIP_CFGIO 1 \
                --set spl.boot.WATCHDOG_ENABLE 0 \
                --set spl.debug.DEBUG_MEMORY_ADDR 0xfffffd00 \
                --set spl.debug.DEBUG_MEMORY_SIZE 0x200 \
                --set spl.debug.DEBUG_MEMORY_WRITE 0 \
                --set spl.debug.HARDWARE_DIAGNOSTIC 0 \
                --set spl.debug.SEMIHOSTING 0 \
                --set spl.debug.SKIP_SDRAM 0 \
                --set spl.performance.SERIAL_SUPPORT 1 \
                --set spl.reset_assert.DMA 0 \
                --set spl.reset_assert.GPIO0 0 \
                --set spl.reset_assert.GPIO1 0 \
                --set spl.reset_assert.GPIO2 0 \
                --set spl.reset_assert.L4WD1 0 \
                --set spl.reset_assert.OSC1TIMER1 0 \
                --set spl.reset_assert.SDR 0 \
                --set spl.reset_assert.SPTIMER0 0 \
                --set spl.reset_assert.SPTIMER1 0 \
                --set spl.warm_reset_handshake.ETR 1 \
                --set spl.warm_reset_handshake.FPGA 1 \
                --set spl.warm_reset_handshake.SDRAM 0 \
                --set spl.boot.SDRAM_SCRUBBING 1 \
                --set spl.boot.SDRAM_SCRUB_BOOT_REGION_START 0x1000000 \
                --set spl.boot.SDRAM_SCRUB_BOOT_REGION_END 0x3000000 \
                --set spl.boot.SDRAM_SCRUB_REMAIN_REGION 1
make -C $(PRELOADER_DIR)
```

Preloader more info

- SoC EDS Users guide – Chapter 7
  - https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_soc_eds.pdf
- Rocketboards
  - http://www.rocketboards.org/foswiki/Documentation/PreloaderUbootCustomization131
  - http://www.rocketboards.org/foswiki/Documentation/GSRD141Preloader

u-boot Generation

u-boot

- Linux workshop – WS2
- Altera SoC 2 day training
- u-boot can be downloaded from GitHub.org…
  - https://github.com/altera-opensource
  - http://www.rocketboards.org/foswiki/Documentation/GitGettingSTarted
- …or built along with the preloader
  - Type “make uboot” in the preloader directory and u-boot will be created

Working with GSRD SD cards

Creating an SD card from GSRD on Linux Host

The steps required to create the SD card for the Cyclone V Development board are:

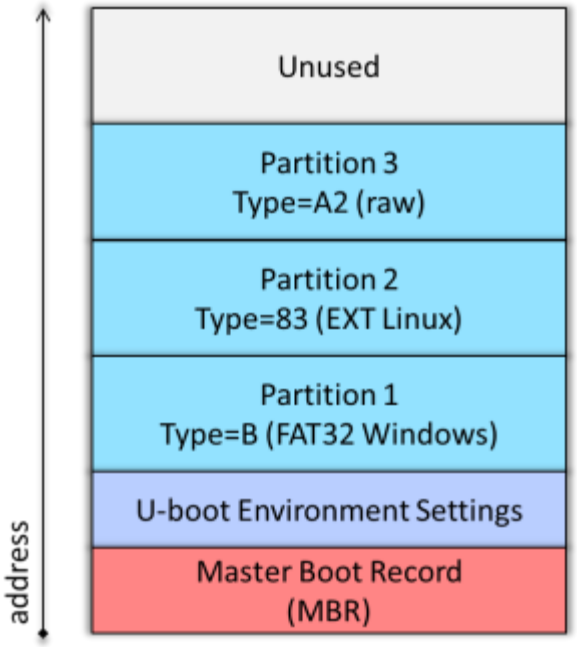
1. Download the GSRD release binaries
   - http://releases.rocketboards.org/release/2014.12/gsrd/bin
2. Extract the compressed Linux SD card image from archive
   - $ tar -xzf linux-socfpga-gsrd-14.1-cv-bin.tar.gz
3. Expand the compressed Linux SD card image
   - $ gunzip linux-socfpga-gsrd-14.1-cv-bin/sd_image_cyclone5.bin.gz
4. Determine the device associated with the SD card on the host by running the following command before and after inserting the card in the reader:
   - $ cat /proc/partitions
   - Let's assume it is /dev/sdx.
5. Use dd utility to write the SD image to the SD card:
   - $ sudo dd if=linux-socfpga-gsrd-14.1-cv-bin/sd_image_cyclone5.bin of=/dev/sdx bs=1M
     - Note we are using sudo so we can access the device node

Creating an SD card from GSRD on Windows Host

- Download and unzip the sd-image
- Download Win32DiskImager from http://sourceforge.net/projects/win32diskimager/
- Insert an micro-SD card in a microSD->USB adapter
- a new drive letter will show up, assuming F:
- run Win32DiskImager
- select the GSRD sd_image for you board
- click 'write‘ and confirm 'yes'
- wait for the completion
- eject the micro-SD card

Partition Layout and Contents

- http://www.rocketboards.org/foswiki/Documentation/GSRD141SdCard



Location	File Name	Description
Partition 1	socfpga.dtb	Device Tree Blob file
	soc_system.rbf	FPGA configuration file
	u-boot.scr	u-boot script for configuring FPGA
	zImage	Compressed Linux kernel image file
Partition 2	various	Linux root filesystem
Partition 3	n/a	Preloader image
	n/a	u-boot image

Updating an SD Card Linux

The following table presents how each item can be updated individually. Replace "sdx" in the command below with the device name of the SD card on your host system. You can find out the device name by running “$ cat /proc/partitions” before and after plugging in the card reader into the host.

File	Update Procedure

zImage	Mount /dev/sdx1 (FAT) on the host machine and update files accordingly:
	$ sudo mkdir sdcard
	$ sudo mount /dev/sdx1 sdcard/
	$ sudo cp <file_name> sdcard/
	$ sudo umount sdcard

soc_system.rbf	
soc_system.dtb	
u-boot.scr	
preloader-mkpimage.bin	$ sudo dd if=preloader-mkpimage.bin of=/dev/sdx3 bs=64k seek=0
u-boot-socfpga_cyclone5.img	$ sudo dd if=u-boot-socfpga_cyclone5.img of=/dev/sdx3 bs=64k seek=4
root filesystem	Mount /dev/sdx2 (ext3 FS) on the host machine and update files accordingly

Updating an SD Card on Windows or Linux Hosts

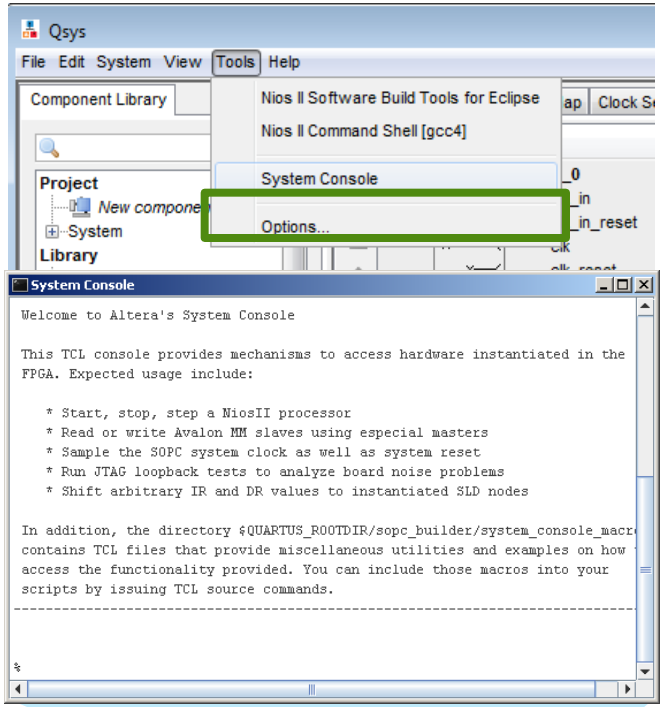
- SD Card Boot Utility
  - Useful for updating an existing sd card image
  - SoC EDS user guide – Chapter 11
  - https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_soc_eds.pdf

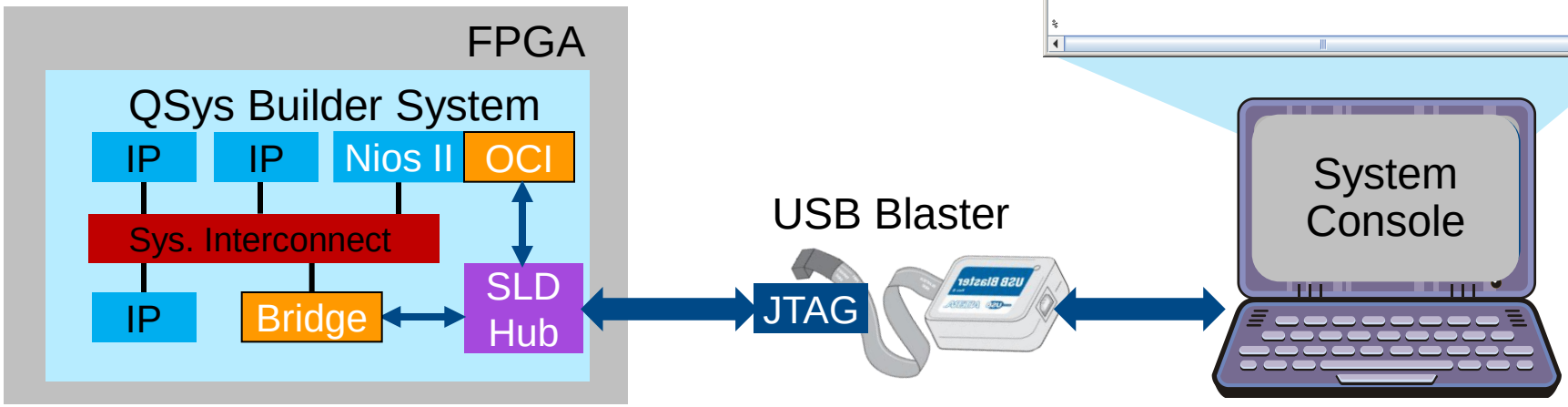
```
$ alt-boot-disk-util
Altera Boot Disk Utility
Copyright (C) 1991-2014 Altera Corporation

Usage:
#write preloader to disk
    alt-boot-disk-util -p preloader -a write disk_file

#write bootloader to disk
    alt-boot-disk-util -b bootloader -a write disk_file

#write BOOTloader and PREloader to disk
    alt-boot-disk-util -p preloader -b bootloader -a write disk_file

#write BOOTloader and PREloader to disk drive 'E'
    alt-boot-disk-util -p preloader -b bootloader -a write -d E
```

System Console

System Console

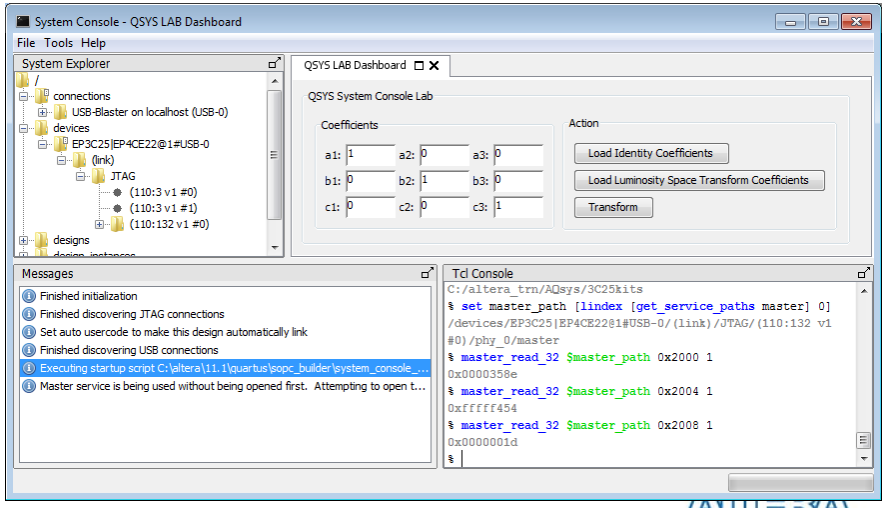
- Provides direct system access at runtime from TCL command line interface
- No processor code to write
  - Access via Nios® II processor, JTAG bridge, JTAG UART or custom Agent
  - Supports an Avalon-MM or Avalon-ST Master
  - Use for board bring-up, debug, diagnostics, system profiling and configuration





What is System Console?

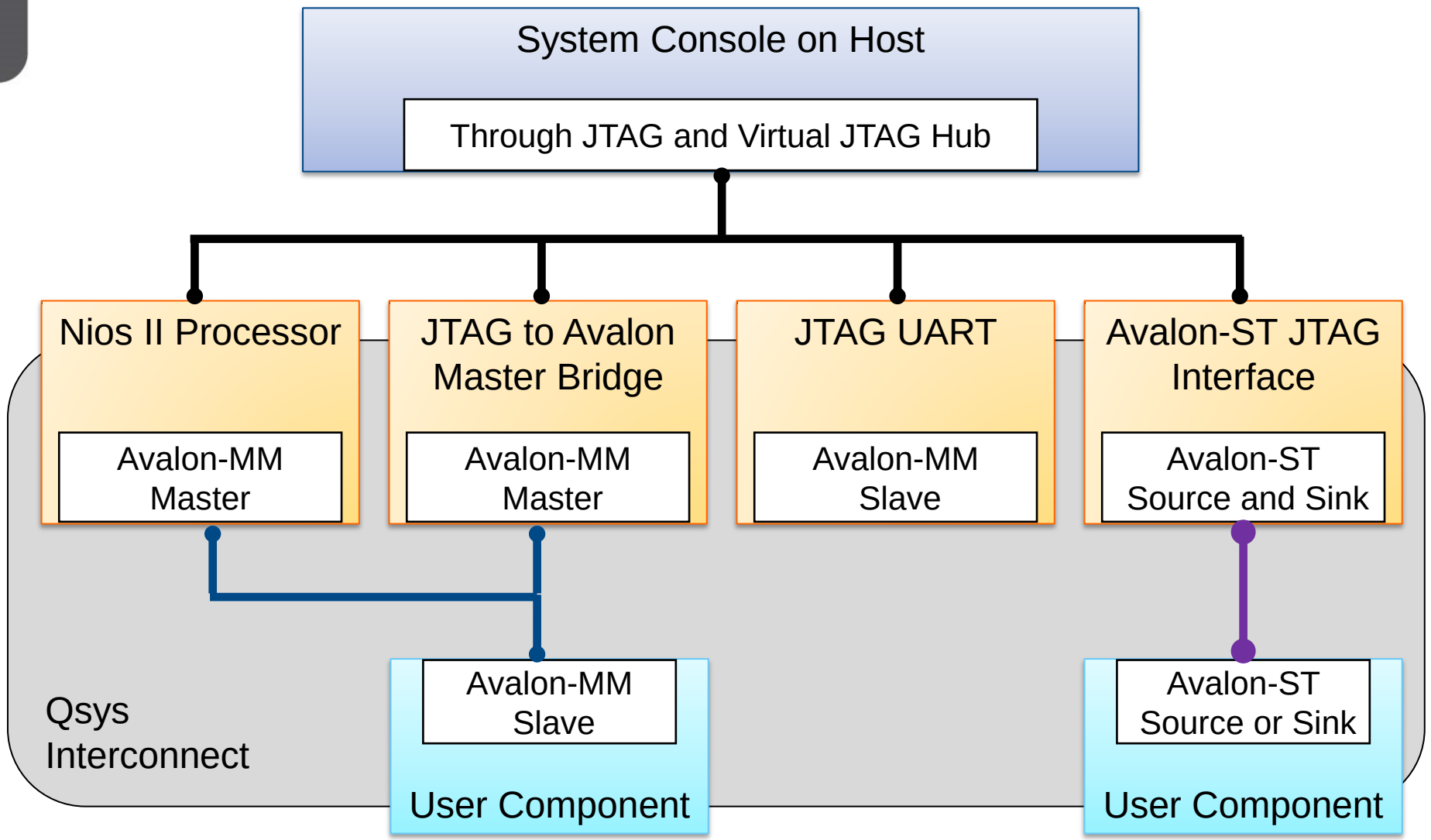
- Quick system-level debug of Qsys systems
  - Interactive Tcl Console
    - Opens as a separate window
    - Launched from the Embedded Command Shell
      - “system-console”
    - Dashboard components available.
  - Debug over various communication channels
    - JTAG or TCP/IP
  - Read or write memory mapped components
  - Sink or source streaming data
  - No processor required



Usage Examples

- System-level debug
  - Board bring-up and interface testing
  - System clock, reset and JTAG chain validity testing
  - Qsys component functionality testing
  - Loopback testing of Avalon Streaming interfaces
  - Inject test vectors, retrieve responses
  - Peek and Poke at the address map.

System Console Interfaces



Usage Flow – Summary

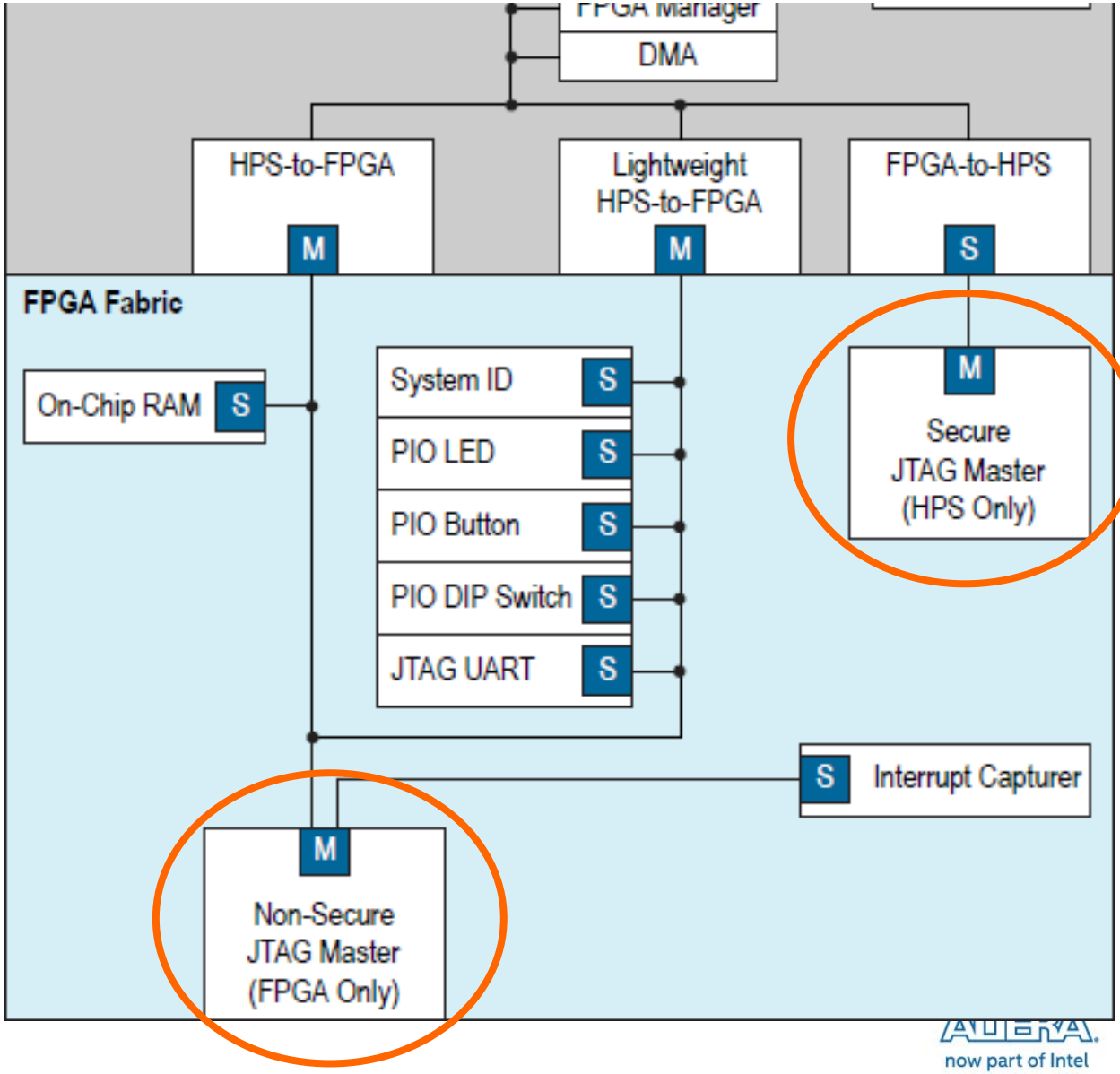
1. Add required component to Qsys
2. Connect board and program FPGA
3. Launch System Console
4. Locate and open service path
5. Perform desired operation(s) with the service
6. Close the service

Complete master write and read example script

```
set m_path [lindex \
    [get_service_paths master] 0]
open_service master $m_path


master_write_memory $m_path 0x2000 \
    [list 0x01 0x02]
master_read_memory $m_path 0x2000 2


close_service master $m_path
```

JTAG Masters in GHRD

- GHRD has multiple JTAG Masters
  - To HPS
  - FPGA slaves
  - To SDRAM (optional)



System Console Resources

- System Console home page
  - https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_system_console.pdf
- On alterawiki.com
  - http://www.alterawiki.com/wiki/System_Console
- Online training courses available.

Creating FPGA specific header files for software development

Creating FPGA-Specific Header Files

“sopc-create-header-files <project>.sopcinfo”

- Creates a header file for each master in the Qsys system.
- Use this in your software flow to keep everything updated.
- Call from an embedded command shell or in your own make file.
- Hardware changes made in Qsys get generated into the SOCINFO file and this utility extracts that information into C header files for your software project.
- sopc-create-header-files is installed in with the ACDS tool chain.

Labs Overview

LABs Overview

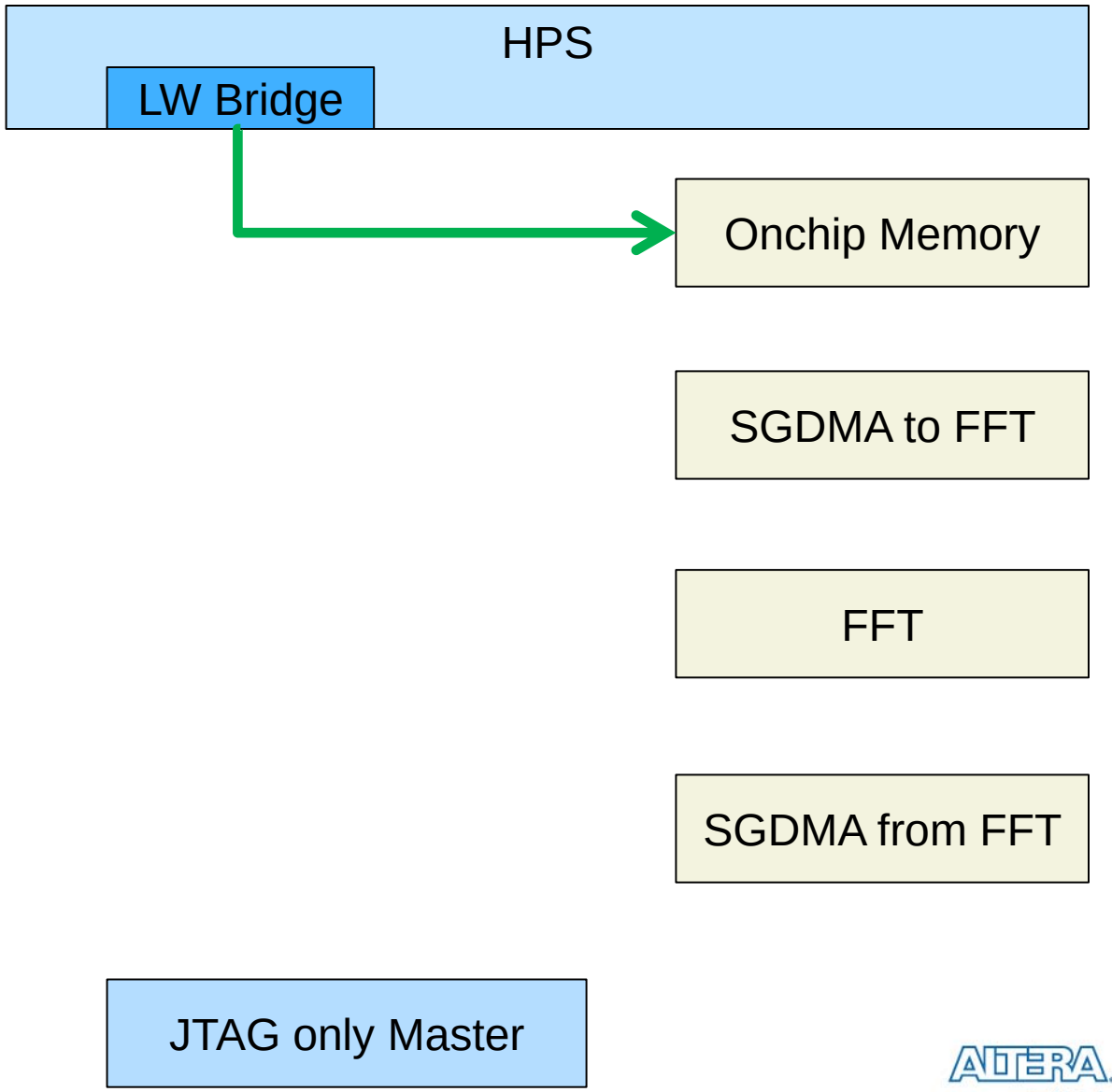
- FPGA hardware already built
  - Based on GHRD
  - Added FFT megacore
  - Added 2 SGDMA to move the data in and out of the FFT
  - Added on chip memory to make transfers easy.
  - Self contained in its own Qsys subsystem.
  - LW bridge and JTAG Master connect to all FFT slaves
- Need Quartus, Cyclone V devices and SoC EDS loaded on your machine.
- No licenses are required for these LABs

- *If you recompile the hardware in Quartus you will need a full Quartus license or 60 day evaluation license.

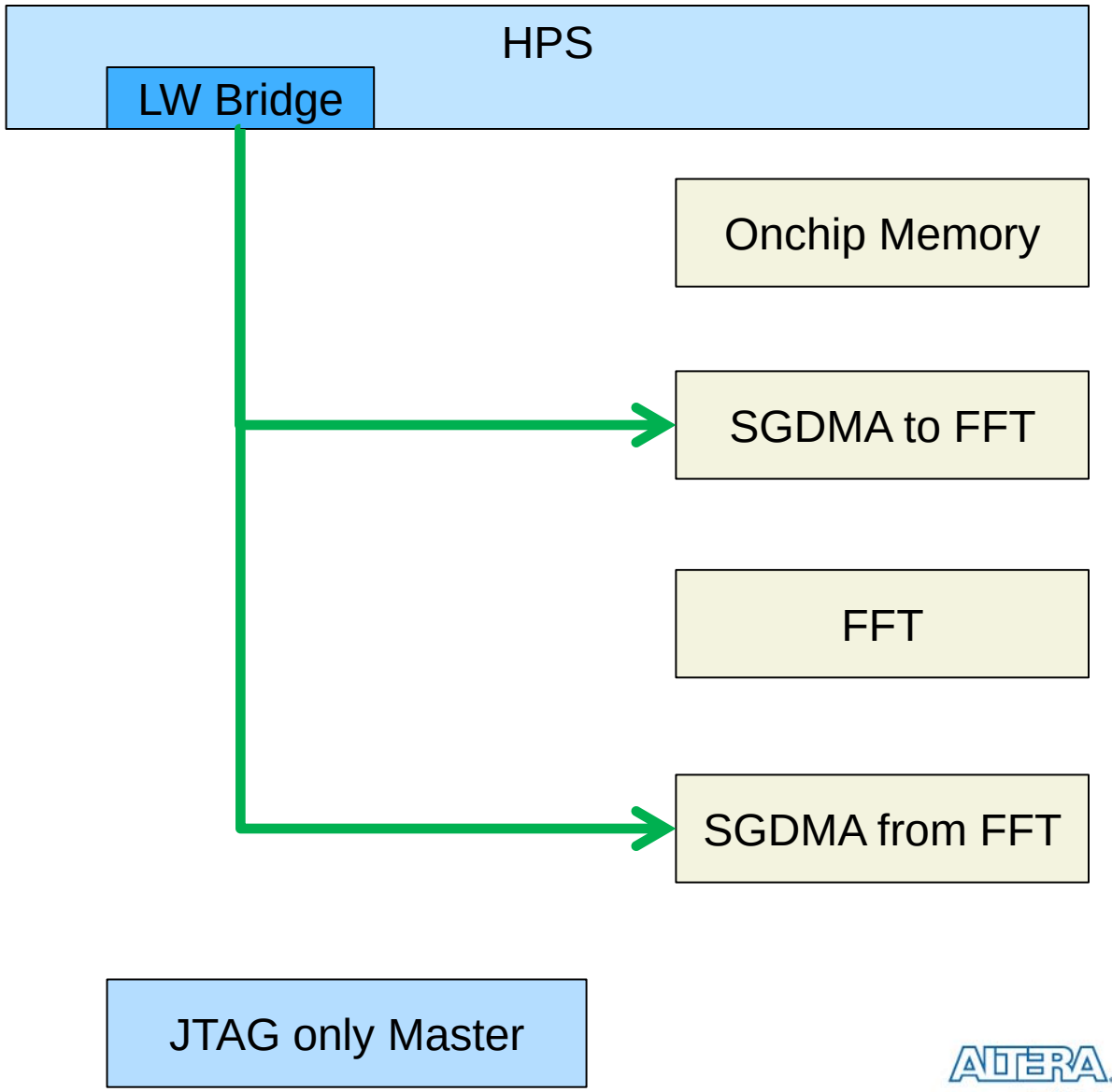
FFT Implementation Details

- FFT Component is Composed of two items
  - The FFT megacore
  - An adapter to wrap it with standard Avalon-ST interfaces
- Inputs are 16-bit real and 16-bit imaginary so it would make a nice 32-bit word.
  - They could be larger, but then you would just make the real and imaginary their own 32-bit words.
- Output is 24-bit of real and 24-bit of imaginary data
  - So map it into 64 bits (2 x 32 bit words);

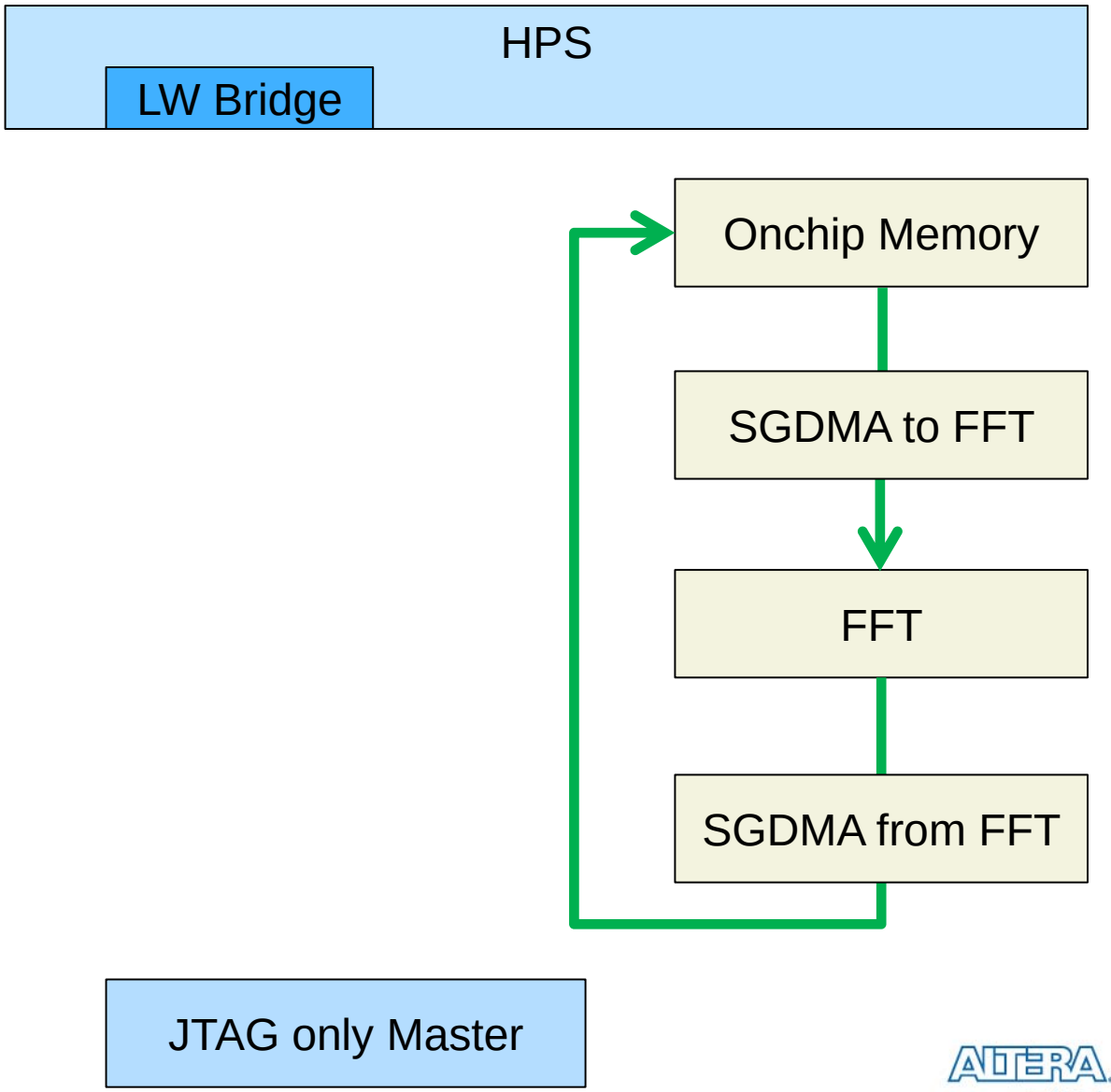
1- Processor Writes wave form to Onchip Memory.



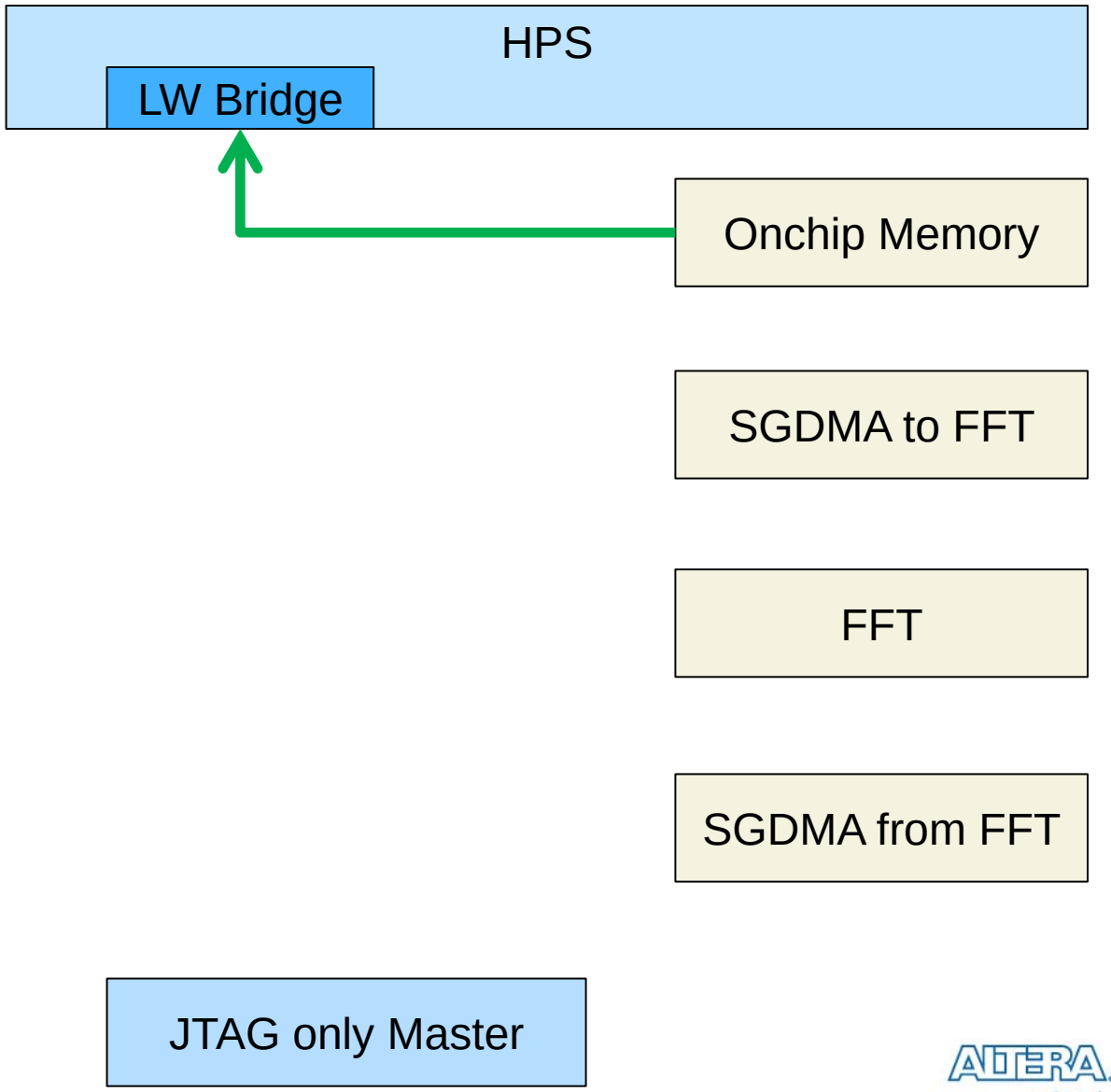
1- Processor Writes wave form to Onchip Memory.
2- Processor Writes Descriptor to SGDMAs to start data flowing.



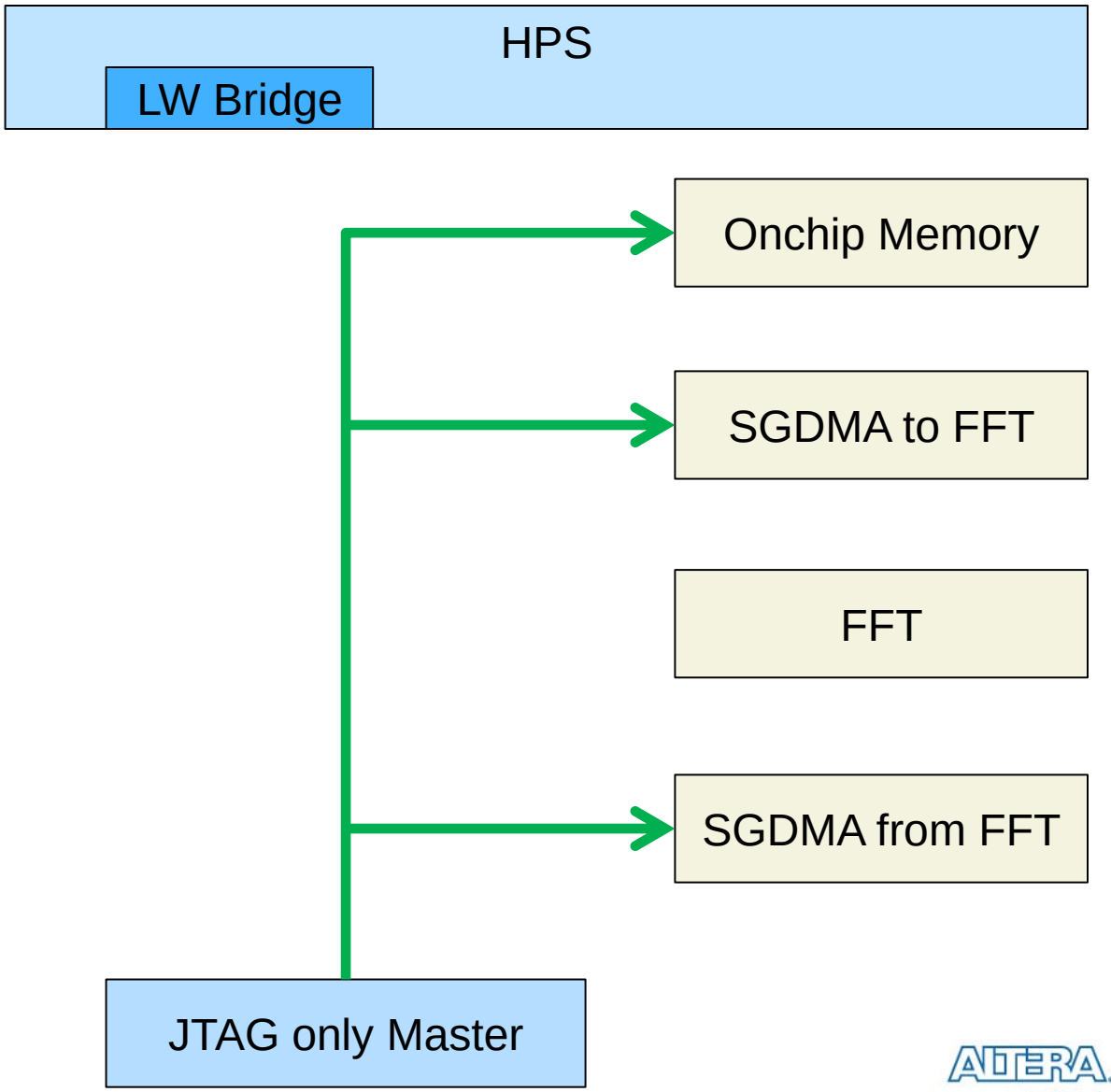
1- Processor Writes wave form to Onchip Memory.
2- Processor Writes Descriptor to SGDMAs to start data flowing.
3-SGDMAs move data through the FFT.



1- Processor Writes wave form to Onchip Memory.
2- Processor Writes Descriptor to SGDMAs to start data flowing.
3-SGDMAs move data through the FFT.
4-Processor reads the FFT data back.



JTAG Master also has access to all theses peripherals so System Console can run the FFT



LABs

- LAB 1 – Generate and Build a Preloader
- LAB 2 – System Console, FFT Hardware Manipulation
- LAB 3 – BareMetal Application, FFT Hardware Manipulation
- LAB 4 – Linux Application, FFT Hardware Manipulation

LAB 1 – Generate and Build a Preloader

- Load SD card with a default image.
- Build Preloader from prebuilt hardware.
- Load new Preloader into SD Card
- Boot target with new Preloader
- Restore original Preloader

LAB 2 – System Console, FFT Hardware Manipulation

Use the FPGA JTAG Master to load the input sample waveforms, trigger the SGDMAs to process the FFT and retrieve the resultant waveforms (no processor involvement required)

- Open system_console
- Load Design SOF
- Run fft dashboard script
- Observe controls and waveforms in dashboard
- This essentially validates the hardware

LAB 3 – BareMetal Application, FFT Hardware Manipulation

- Review code
- Use the alt_read_word, alt_write_word to insure the data does not get cached by the processor
- Compile in an embedded_command shell
  - Paths to the hwlibs, socal and compilers
- Load code onto SD card and run from u-boot
- Program the Preloader from Lab 1 and the bare metal application into 0xA2 partition to observe auto boot into the bare metal application.

LAB 4 – Linux Application, FFT Hardware Manipulation

- Linux requires that a driver must be present to talk to hardware
- There are at least three drivers that can be used.
  - /dev/mem, ready to go out of the box
  - uio driver, some assembly required
  - Custom written driver, you own it
- Not a linux class so we use the simplest /dev/mem
- Linux also uses the MMU so all address are virtual
- Linux drivers discussed in detail in Developing Drivers for Altera SoC Linux Workshop

LAB 4 – Linux Application, FFT Hardware Manipulation

- Build the linux application by running the make file
- Copy the fpga_fft application over to the target
- Run the application on the target
- Observe the FFT input and output data graphically through the target based web server:
  - Two files fft.sh and MyGraphTest1_2.js facilitate this on the target
  - fft.sh is the initial script that is run when you type the below in your browser
    - <ipaddress>/fft.sh
  - MyGraphTest1_2.js is a JavaScript which plots the FFT input and output and runs in the browser
  - When fpga_fft is runs, it produces output files with the input samples and output result values of the FFT.
  - This JavaScript code reads it in and plots the graph.

Thank You