# ACHILLES

## Instant-DevKit
## Arria® 10 SoC SoM


# Reference Manual

**First Release : 2016-12**
Web: www.reflexces.com
Support: https://support.reflexces.com

**Rev 2.9**
**2019-08**


**REFLEX CES Europe, Japan**
Email : sales@reflexces.com
**REFLEX CES Americas / Asia Pacific**
Email : salesusa@reflexces.com

# Table of Contents

RECORD OF REVISION ..... 5

PROPRIETARY NOTICE ..... 7

1. PREFACE ..... 8

1.1 About This Manual ..... 8
1.2 Contact Information ..... 9
1.3 Abbreviations_Figures_Tables ..... 10
1.3.1 Abbreviations ..... 10
1.3.2 List of Figures ..... 11
1.3.3 List of Tables ..... 12
1.4 Additional Reading ..... 13
1.5 Electrostatic Discharge ..... 14

2. PACKAGE CONTENTS ..... 15

2.1 USB Key Package Contents ..... 17
2.2 Board Package Content ..... 18
2.3 Purpose of the Arria® 10 FPGA SoC 660KLEs ..... 19

3. PRODUCT OVERVIEW ..... 20

3.1 Block Diagram ..... 20
3.1.1 Arria® 10 SoC SoM Boards ..... 20
3.1.2 Achilles Starter Board ..... 22
3.1.3 PCIe Carrier Board ..... 22
3.2 Board Layout ..... 23
3.2.1 Achilles Starter Board ..... 24
3.2.2 PCIe Carrier Board ..... 24
3.2.3 Feature List and Details ..... 25
3.3 SoM A10 SoC Lite Version ..... 26
3.3.1 Overview ..... 26
3.3.2 Feature List and Details ..... 27

4. PRODUCT SPECIFICATIONS ..... 28

4.1 Mechanical Description ..... 28
4.2 Electrical Specification ..... 30
4.2.1 Power Input ..... 30
4.2.2 Voltage Rails ..... 31
4.2.3 Power Consumption ..... 31
4.3 Thermal Specifications ..... 35
4.4 Regulatory Compliance ..... 36
4.4.1 Electromagnetic Compatibility ..... 36
4.4.2 Safety ..... 36

4.5 Reliability ..... 37
5. BOARD INSTALLATION ..... 38
6. BOARD CONFIGURATION ..... 40
6.1 Tools Configuration ..... 40
6.2 Innovative Graphical User Interface (GUI) ..... 40
6.3 Board Programming Methods ..... 41
7. HARDWARE DESCRIPTION ..... 43
7.1 Signal Pinouts General Description ..... 43
7.2 Clocks Description ..... 44
7.2.1 Clock Tree ..... 44
7.2.2 Clocks Assignments ..... 45
7.2.3 Programmable Clock Generator ..... 46
7.3 Power Management ..... 47
7.4 SoM Connectors ..... 48
7.5 Board LEDs ..... 48
7.5.1 Status LEDs ..... 48
7.5.2 User LEDs ..... 49
7.6 User Dipswitch ..... 50
7.7 MAX® 10 FPGA connections ..... 51
7.8 PCIe-Specific Interfaces ..... 53
7.9 DDR4 FPGA interface ..... 56
7.10 DDR4 HPS Interface ..... 58
7.11 FX3 interface ..... 61
7.12 FMC HPC TOP interface ..... 62
7.12.1 FMC Capabilities ..... 62
7.12.2 FMC Mechanical Capabilities ..... 62
7.12.3 FMC Modules Capabilities ..... 63
7.13 10/100/1000 Ethernet Copper Solution ..... 64
7.14 MAX® 10 System Controller ..... 66
7.14.1 Architecture ..... 66
8. HPS DEVELOPMENT ..... 72
8.1 Tools ..... 72
8.2 HPS Peripherals ..... 73
8.2.1 Supported Features ..... 73
8.2.2 Achilles DevKit HPS Peripherals ..... 75
8.3 HPS Software Architecture ..... 76
8.3.1 BootROM ..... 76
8.3.2 U-Boot ..... 76
8.3.3 Linux Kernel ..... 76
8.3.4 Linux Root File Systems ..... 76

8.4 HPS Boot Flow ..... 78
8.4.1 Achilles DevKit Default Boot Flow ..... 78
8.4.2 Achilles Devkit Factory Boot Flow ..... 79
8.5 eMMC Layout ..... 81
8.6 HPS Folder Contents ..... 82
8.6.1 Autobuild ..... 82
8.6.2 Factory ..... 82
8.6.3 Kernel ..... 82
8.6.4 Rootfs ..... 83
8.6.5 Test ..... 83
8.6.6 Tools ..... 84
8.6.7 U-Boot ..... 84
8.6.8 ReleaseNotes.txt ..... 84
8.7 Build Flow ..... 85
8.7.1 Shell environment mandatory requirements ..... 86
8.7.2 Automatic Build ..... 86
8.7.3 Root File Systems ..... 90
8.7.4 U-Boot Unitary Build ..... 90
8.7.5 Kernel Unitary Build ..... 97
8.8 Board Power Up ..... 99
8.8.1 BSEL Configuration ..... 99
8.8.2 Serial Connection ..... 101
8.8.3 Ethernet Connection ..... 101
8.9 HPS Test ..... 103
8.10 Development Process ..... 104
8.10.1 Loading U-Boot in OCRAM ..... 104
8.10.2 Loading Kernel through TFTP ..... 105
8.10.3 Mounted NFS Root File System ..... 108
8.11 How to Restore / Update eMMC Partition Content ..... 110
8.11.1 Restoring eMMC Partitions ..... 110
8.11.2 Updating an eMMC partition ..... 111
9. COMPONENT REFERENCES ..... 113
10. TROUBLESHOOTING ..... 115

# RECORD OF REVISION

| Date | Doc Version | Board Version | Description |
|---|---|---|---|
| December 2016 | 1.0 | 1.0 | Preliminary customer delivery |
| January 2017 | 1.1 | 1.0 | Add details on “Achilles devkit factory boot flow” paragraph.<br>Fix HPS Autobuild default and factory build flow paragraphs.<br>Edit “HPS content folder” section: adding new daemon to monitor HPS in the Achilles devkit GUI. |
| February 2017 | 1.2 | 1.0 | Edit HPS chapters to add / modify information concerning:<br>- Host software environment<br>- Boot flow<br>- Serial and Ethernet connections<br>- Restoring / update eMMC partitions<br>- External toolchain for automatic build |
| May 2017 | 1.3 | 1.0 | Updated board architecture description<br>Add MAX® 10 System Controller description |
| June 2017 | 1.4 | 1.0 | Edit chapters concerning bridges, as we removed our driver and fitted Altera’s.<br>Changed screenshot of bsp-editor to match these bridge changes. |
| August 2017 | 2.0 | 2.0 | Added support for 2.0 board revision |
| September 2017 | 2.1 | 2.0 | Fix minor pinout issues |
| October 2017 | 2.2 | 2.0 | Editorial updating |
| January 2018 | 2.3 | 2.0 | Updated new linaro-based rootfs changes,<br>Kernel building update<br>Editorial updating<br>SoM A10 SoC Lite/Indus/Turbo Specificities |
| July 2018 | 2.4 | 2.0 | SoM A10 SoC Indus/Turbo/Lite block diagrams are defined<br>Meaning of “som_pg” signal completed<br>“Sales Germany” is added.<br>Adding Linux restriction for automatic / U-Boot / kernel build<br>Fixing SSH connection with root access for Linaro-based root file system<br>A warning about transceiver recalibration is added in chapter 6.3. |
| September 2018 | 2.5 | 2.0 | P42, Fig.24: signals CLK_HPS_DDR4 and CLK_FPGA_DDR4 (single ended signals) updating. |
| November 2018 | 2.6 | 2.0 | Updating:<br>- MAX® 10 and Arria® 10 with the ® trademark<br>- 7.2.3 section: The clock generator can be programmed in-circuit by **the MAX® 10** through the I2C serial interface. |

| Date | Doc Version | Board Version | Description |
|---|---|---|---|
| | | | - Figure 5: Module Block diagram for the Turbo, Figure 6: Module Block diagram for the Indus version, Figure 7: Module Block diagram for the Lite version and Figure 18: Connector view updated with 34 LVDS.<br>- 3.2.3 section: 34 LVDS bidirectional |
| December 2018 | 2.7 | 2.0 | Update HPS content after SoCEDS migration to version 18.1 |
| July 2019 | 2.8 | 2.0 | Updates:<br>_Board Layout with bottom view<br>_BootSEL switches: switch positions and location on the board<br>_MSEL1switch: switch location on the board and update of "Board programming Method" section<br>_Board LEDs: LED location, Table 5: FPGA LEDs status and Table 6: Running User LEDs.<br>_User dipswitch section<br>_Sales US and Japan removed<br>_Build flow instructions<br>- Figure 24: Correct clock name and add missing clocks<br>- Figure 30: remove the Push button<br>- Figure 31: Correct Lane distribution<br>- 7.8: Add PCIe Lane distribution table<br>- 7.12.1: Correct FMC HPC capabilities<br>- 9: Component references added |
| August 2019 | 2.9 | 2.0 | Update HPS content due to following HPS modifications:<br>- Fix conflict between Buildroot and SoCEDS in autobuild<br>- Fix U-Boot 2GB RAM limitation<br>- Use only Linaro root file system for default boot flow<br>- Add Linaro toolchain as external tool<br>- Remove the need to source SoCEDS environment file<br>- Add a Factory folder with kernel image integrating RAM file system<br>- Add autobuild script second parameter ("auto" or "manual") for the "factory" target to control partitioning script at boot |

# PROPRIETARY NOTICE

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by REFLEX CES. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. REFLEX CES give all particulars of the product and its use contained in this document in good faith. This document is provided “as is” with no warranties whatsoever, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample.

This document is intended only to assist the reader in the use of the product. REFLEX CES shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product. Nor shall REFLEX CES be liable for infringement of proprietary rights relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

# 1. PREFACE

## 1.1 About This Manual

This document has been written for design managers, system engineers, and designers of ASICs and FPGAs who are evaluating or using the REFLEX CES Instant-DevKit Achilles Arria® 10 Soc SoM.
Images in this document are for illustrative purposes only. They are not contractual and are subject to change.

This reference manual gives customers:

- Package Content description,
- Features and parameters of the board.
- Configurations and specifications of the board.

Throughout the manual user must pay particular attention to the **WARNINGS**, **CAUTIONS,** and *NOTES*.

Their intentions are as follows:

**WARNING:** to call attention to methods and procedures which must be followed precisely to prevent form injury to operators.

**CAUTION**: to call attention to methods and procedures which must be followed to prevent from damage to equipment.

*NOTE: To call attention to methods or information which allow a better understanding.*

## 1.2 Contact Information

**Corporate Headquarters:**

REFLEX CES
2 Rue du Gévaudan
91047 EVRY Cedex
France
Phone: +33 (0)169870255
Fax: +33 (0)164972859
www.reflexces.com

**Sales Europe, Japan**

For sales questions, please contact: sales@reflexces.com.

**Sales Americas, Asia Pacific**

For sales questions, please contact: salesusa@reflexces.com.

**Support**

For technical questions, please contact REFLEX support at https://support.reflexces.com/

**Quality/Reliability**

For reliability questions, please contact REFLEX quality service at certification@reflexces.com.

## 1.3 Abbreviations_Figures_Tables

### 1.3.1 Abbreviations

| ABBREVIATION | MEANING |
|---|---|
| BMC | Board Management Controller |
| BOM | Bill Of Materials |
| CAUI | 100 Gigabit Attachment Unit Interface |
| CPLD | Complex Programmable Logic Device |
| CPPI | 100 Gigabit Parallel Physical Interface |
| CvP | Configuration via Protocol |
| DDD | Detailed Design Document |
| DDR | Double Data Rate |
| DDR3 | Double Data Rate 3rd generation |
| DDR4 | Double Data Rate 4th generation |
| ECC | Error Correction Coding |
| eMMC | Embedded MultiMedia Card |
| DSP | Digital Signal Processor |
| EDR | Enhanced Data Rate |
| EPLD | Erasable Programmable Logic Device |
| ESD | ElectroStatic Discharge |
| FPGA | Field Programmable Gate Array |
| FMC | FPGA Mezzanine Card |
| GbE | Gigabit Ethernet |
| GPIO | General Purpose Input Output |
| HIP | Hard IP |
| HPC | High Pin Count Connector with 400 pins |
| HPS | Hard Processor System |
| I2C | Inter-Integrated Circuit |
| IPMI | Intelligent Platform Management Interface |
| JTAG | Joint Test Action Group |
| LED | Light Emitting Diode |
| LPC | Low Pin Count |
| LVDS | Low Voltage Differential Signaling |
| N/A | Not Applicable |
| NC | Not Connected |
| NVME | Non-Volatile Memory Express |
| OTG | On The Go |
| PCB | Printed Circuit Board |
| PCIe | PCI Express |
| PCS | Printed Circuit Specifications |
| PLA | Programmable Logic Array |

| ABBREVIATION | MEANING |
|---|---|
| PPS | Pulse Per Second |
| PMbus | Power Management Bus |
| QSFP | Quad Small Form-factor Pluggable |
| QDR | Quad Data Rate |
| QSPI | Quad Serial Peripheral Interface |
| RTC | Real Time Clock |
| SFP | Small Form-factor Pluggable |
| SDM | Secure Device Manager |
| SDRAM | Synchronous Dynamic Random-Access Memory |
| SOM | System On Module |
| TBC | To Be Confirmed |
| TBD | To Be Defined |
| UART | Universal Asynchronous Receiver Transmitter |
| USB | Universal Serial Bus |

*Table 1: Table of abbreviations*

### 1.3.2 List of Figures

Figure 1: ESD protection equipment....14
Figure 2: DevKit Content....15
Figure 3: ESD Shielded bag and ESD prevention....16
Figure 4: Board Package Content....18
Figure 5: Module Block diagram for the Turbo....20
Figure 6: Module Block diagram for the Indus version....21
Figure 7: Module Block diagram for the Lite version....21
Figure 8: Achilles Starter Board Block Diagram....22
Figure 9: PCIe Carrier Board Block Diagram....22
Figure 10: Layout and Components....23
Figure 11: SoM A10 SoC module with air cooled system....23
Figure 12: Achilles Starter Board....24
Figure 13: PCIe Carrier Board....24
Figure 14: SoM A10 SoC Lite version with air cooled system....26
Figure 15: Module Mechanical Dimensions....28
Figure 16: Achilles Starter Board Mechanical Dimensions....29
Figure 17: PCIe Carrier board power supplies....30
Figure 18: Connector view....38
Figure 19: Use Case 1....39
Figure 20: Use Case 2....39
Figure 21: Use Case 3....39
Figure 22: Board GUI....40
Figure 23: Board Programming....41
Figure 24: Clock Tree Overview....44
Figure 25: Power Distribution....47
Figure 26: Power Signals to Supervisor....47

Figure 27: LED positions ..... 48
Figure 28: LEDs Design Schematic ..... 49
Figure 29: User dipswitch SW5 location ..... 50
Figure 30: User Dipswitch Design Schematic ..... 50
Figure 31: PCIe-Specific Interfaces ..... 53
Figure 32: PCIe Boot Mode with CvP ..... 55
Figure 33: Mechanical Dimensions of the FMC-HPC card ..... 62
Figure 34 : Synoptic ..... 66
Figure 35 : MAX® 10 block diagram ..... 66
Figure 36 : Reset sequence time chart ..... 67
Figure 37 : UART2AVMM QSYS System ..... 70
Figure 38: HPS Block Diagram ..... 74
Figure 39: Default Boot Flow ..... 78
Figure 40: Factory Boot Flow ..... 80
Figure 41: Global Build Flow ..... 85
Figure 42: U-Boot Build Flow ..... 90
Figure 43: I/O Configuration ..... 92
Figure 44: BSEL location ..... 99

### 1.3.3 List of Tables

Table 1: Signal Descriptions ..... 43
Table 2: Pin assignments for the clocks from oscillators ..... 45
Table 3: PLL On Board Clock Output ..... 45
Table 4: Power LEDs status ..... 48
Table 5: FPGA LEDs status ..... 48
Table 6: Running User LEDs ..... 49
Table 7: User Dipswitch Configuration ..... 50
Table 8: MAX® 10 FPGA Connections ..... 52
Table 9: DDR4 FPGA Connections ..... 57
Table 10: DDR4 HPS Connections ..... 60
Table 11: FX3 Connections ..... 61
Table 12: FMC Power Capabilities ..... 63
Table 13: FMC Interface Pinout Compatibility ..... 63
Table 14: Ethernet Connections ..... 65
Table 15: Register table ..... 69
Table 16: Transaction Packet Format ..... 70
Table 17: Response Packet Format ..... 70
Table 18: Transaction Supported ..... 71
Table 19: Operation Avalon-ST Bytes to Packets Converter Core ..... 71
Table 20: Supported peripherals by HPS on the Achilles DevKit ..... 75
Table 21: Script directories ..... 82
Table 22: Example of SoM Board Implementation References ..... 114




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 1.4 Additional Reading

REFLEX CES periodically updates its documentation. Please contact REFLEX CES Technical Support or check the website at http://www.reflexces.com for current versions.

Please refer to the following links for information:

- *ACHILLES_Starter_Guide:* Describes how to power on the board, how to configure the default dipswitch state, and how to launch the GUI.

  *NOTE: This document is not yet available.*

- *ACHILLES_Content_and_Limitations:* Information on specifications and DevKit limitations.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 1.5 Electrostatic Discharge

Electronic boards are sensitive to electrostatic discharges. It is recommended to work in an electrical healthy environment. As for all electronic devices, it is recommended to use ESD protection equipment when handling the board. We strongly suggest following standard ESD care when touching any part of the board.

Please take care to use ESD equipment when handing the board to avoid damaging the board.


- Boards must be used in a specific premise containing the "ESD protected area" mentioned.

- Place your board on an ESD mat and connect your ESD mat to building ground or "green wire" equipment ground.

- Wear an ESD coat and ESD shoes.

- Wear an ESD bracelet connected to your ESD mat when you insert the board into a computer or server.

*Figure 1: ESD protection equipment*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

# 2. PACKAGE CONTENTS

The Achilles Instant Development Kit includes the following elements:

1. The Arria® 10 SoC SoM Version with its custom heatspreader and heatsink including a fan
2. The Achilles Starter Board to use the Arria® 10 SoC SoM
3. A Power desk adapter (AC/DC), 12V, 108W
4. Four Power cables (US, UK, EU, and JP)
5. A USB standard A-type to Micro USB connector cable, 1.8 m length
6. A USB standard female A-type to Mini USB connector B-type cable
7. A USB standard A-type to Micro USB3.0 cable, 1m length
8. Two Ethernet CAT5E RJ-45 cables
9. A USB Key containing REFLEX innovative software interface (GUI / Windows) with test designs and documentation.

*NOTE: The GUI is only delivered for the purchase of the Achilles Instant Development Kit. Refer to the REFLEX CES website at [http://www.reflexces.com/](http://www.reflexces.com/) for more details.*


*Figure 2: DevKit Content*

The Achilles Instant Development Kit boards delivered in an ESD shielded bag with a desiccant pack to protect the board against ESD and humidity. The bag is sealed with a yellow label to inform user that the Achilles DevKit boards contain components sensitive to ESD.


*Figure 3: ESD Shielded bag and ESD prevention*

## 2.1 USB Key Package Contents

The USB key contains all the documentation for the Achilles Development Kit (Starter Guide, Reference Manual, Schematics, Assembly files and Product Brief) and board test designs.

When using the Achilles Instant DevKit for the first time, you can run the innovative Graphical User Interface to explore the DevKit interfaces, applications and performances.

There is no need to install Quartus II to access the board via JTAG communication. The board is delivered pre-programmed - just plug-in the USB cable and launch the executable file of the GUI to start exploring the board in a few seconds.

Binary files such as jic, sof, and pof files, are provided so that advanced users can revert to the delivery state or perform a factory re-set.

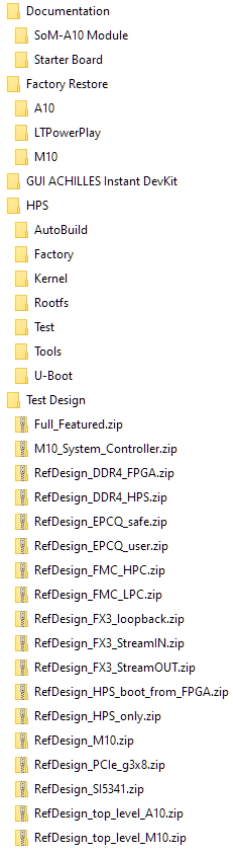
The USB key also contains all the files required to enable test interfaces or to create your own design. The ACHILLES RefDesign_top_level_A10 folder contains a blank Quartus II project, which you can use to build a custom design "from scratch". It does not contain any HDL code, except the top-level port declarations. Pin assignments and timing constraints are provided for reference but must be reviewed and customized to suit your design requirements.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 2.2 Board Package Content

The structure of the board package is shown below:




- **Documentation:**
  - Reference Manual
  - Carrier Design Guide: rules and guidelines for designing you own carrier board
  - SuperPinout File for FMC connectors
  - Schematics for the Starter board only
  - Content & Limitations letter
- **Factory Restore:** Contains files required to recover a delivery state
- **GUI :** Software application to access the FPGA interfaces and monitor performance
- **HPS:** Hard Processor System tarball archive
- **Test Designs:** contains projects to test each interface and start your own design:
  - **Full_Featured :** Starting point for designing your own design. Includes all FPGA interfaces
  - **M10_System_Controller :** Starting point for designing a Max10 (refer to reference manual)
  - **DDR4:** Run the Altera memory test design example
  - **EPCQ:** Manage a SAFE/USER boot mechanism with two Quartus project revisions
  - **FMC HPC/LPC:** Contains two main modules for testing LVDS and transceiver links on connector
  - **FX3 :** Example design for USB3.0 interface
  - **HPS :** FPGA designs to use along with HPS
  - **M10:** Demonstrates the configuration of the SI5341, handles resets and the PMBus
  - **PCIe_g3x8 :** Intel PCIe demo
  - **SI5341:** Demonstrates the configuration of the SI5341
  - **RefDesign_top_level_A10:** Blank Quartus project to start a custom design "from scratch"
  - **RefDesign_top_level_M10:** Blank Quartus project to start a custom design "from scratch"

*Figure 4: Board Package Content*

## 2.3 Purpose of the Arria® 10 FPGA SoC 660KLEs

The Arria® 10 SoC SoM provides developers with the best Out-of-the-Box experience, combining a Best-in-Class compact hardware platform with the most efficient intuitive software environment. Featuring an ARM dual-core Cortex-A9 MPCore and up to 660KLEs of advanced low-power FPGA logic elements, the Arria® 10 SoC incorporates the flexibility and ease of programming of a CPU with the configurability and parallel processing power of an FPGA.

The target markets are as wide as Automotive, Video Broadcast, Machine and Intelligent Vision, Industrial, Military, Test & Measurement and Medical.

The Arria® 10 SoC SoM comes with an intuitive Graphical User Interface (GUI) and offers a powerful hardware platform to develop your own application. An ultra-compact form factor of only 86 x 95 mm (3.4 x 3.8 inches) is ideal for area-constrained applications.

The Arria® 10 module is also a dual FMC carrier board (VITA 57.1). The FMC connector offers the best front-end key solution to define your own custom product. With this connector, you can manage an extensive and flexible IO front-end, which enables quick prototyping and evaluation.

*Important notice:*
*If you notice any missing information, please contact REFLEX CES support (see section 1.2).*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

# 3. PRODUCT OVERVIEW

## 3.1 Block Diagram

### 3.1.1 Arria® 10 SoC SoM Boards

The Arria® 10 SoC SoM Turbo version block diagram is shown below:

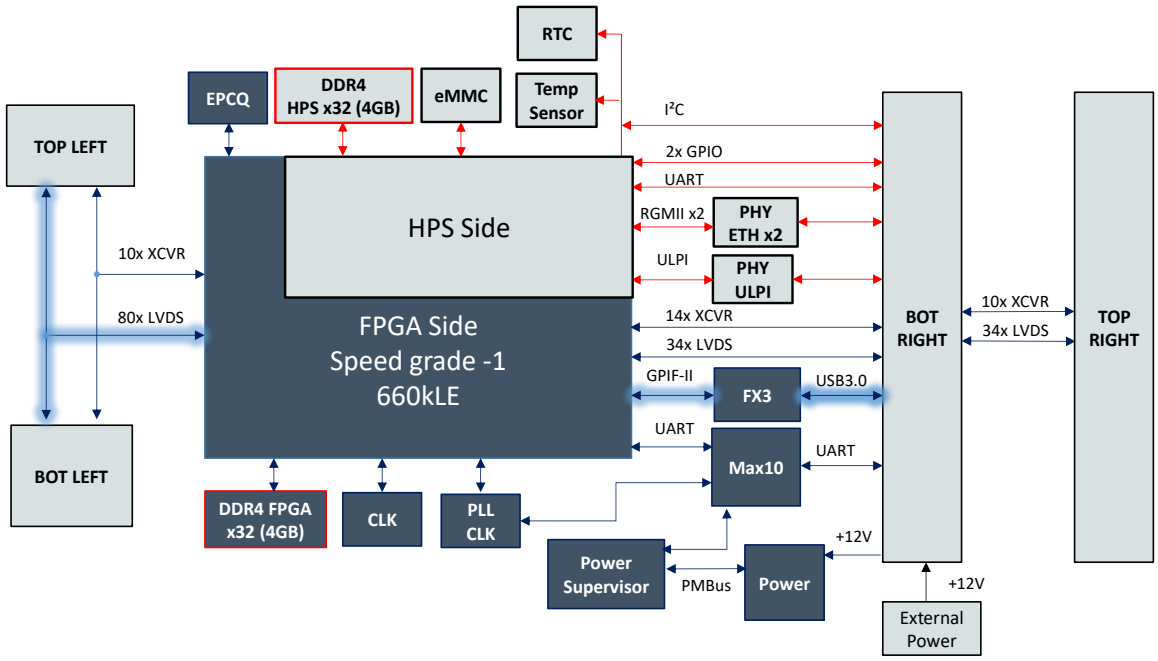



*Figure 5: Module Block diagram for the Turbo*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

The Arria® 10 SoC SoM Indus version block diagram is shown below:

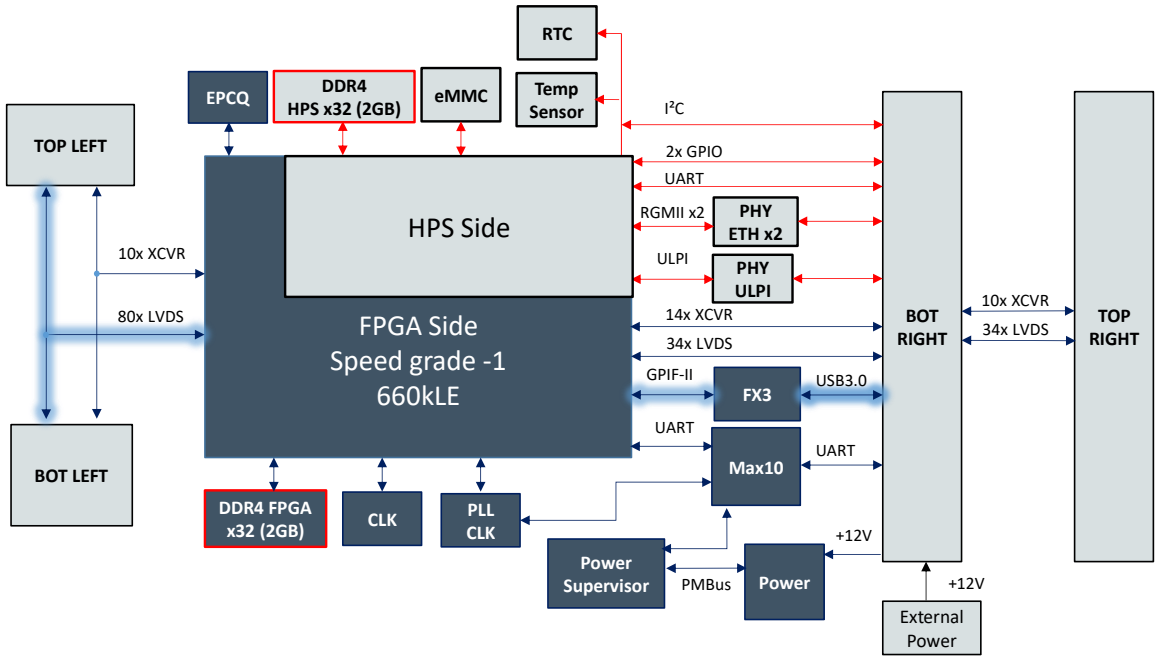



*Figure 6: Module Block diagram for the Indus version*

The Arria® 10 SoC SoM Lite version block diagram is shown below:

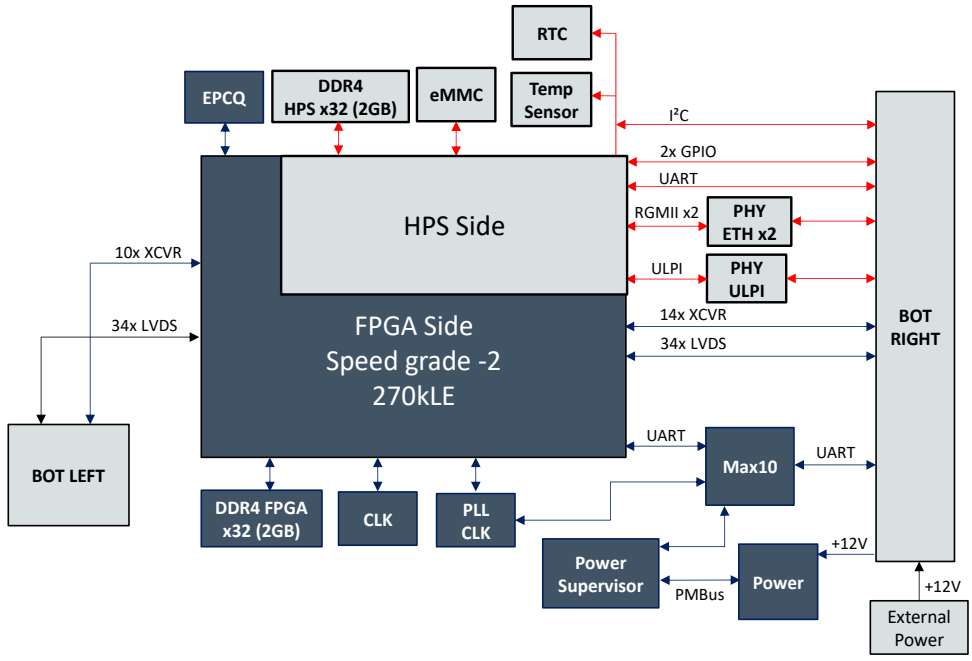



*Figure 7: Module Block diagram for the Lite version*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

### 3.1.2 Achilles Starter Board

The Achilles Starter Board block diagram is shown below:

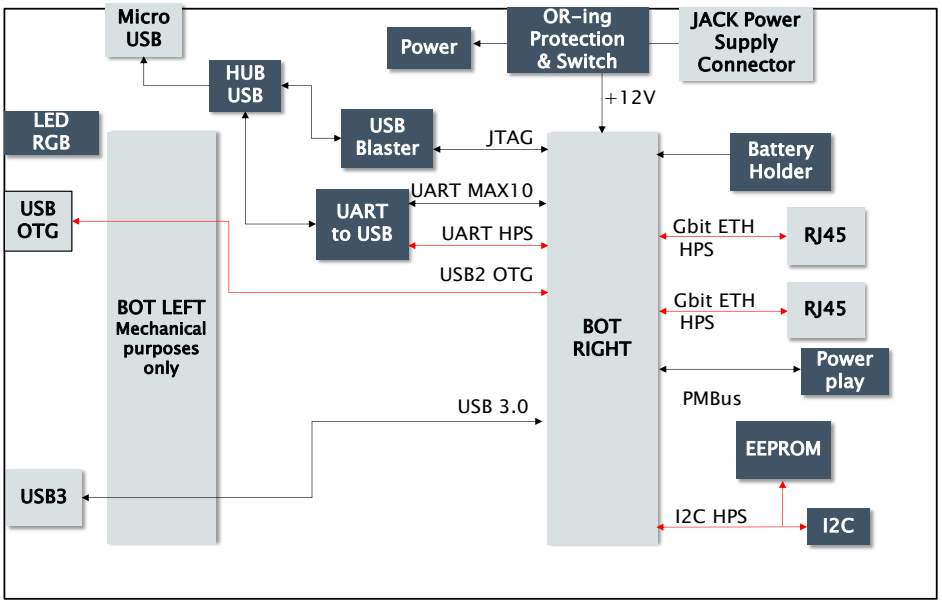



*Figure 8: Achilles Starter Board Block Diagram*

### 3.1.3 PCIe Carrier Board

The PCIe Carrier Board block diagram is shown below:

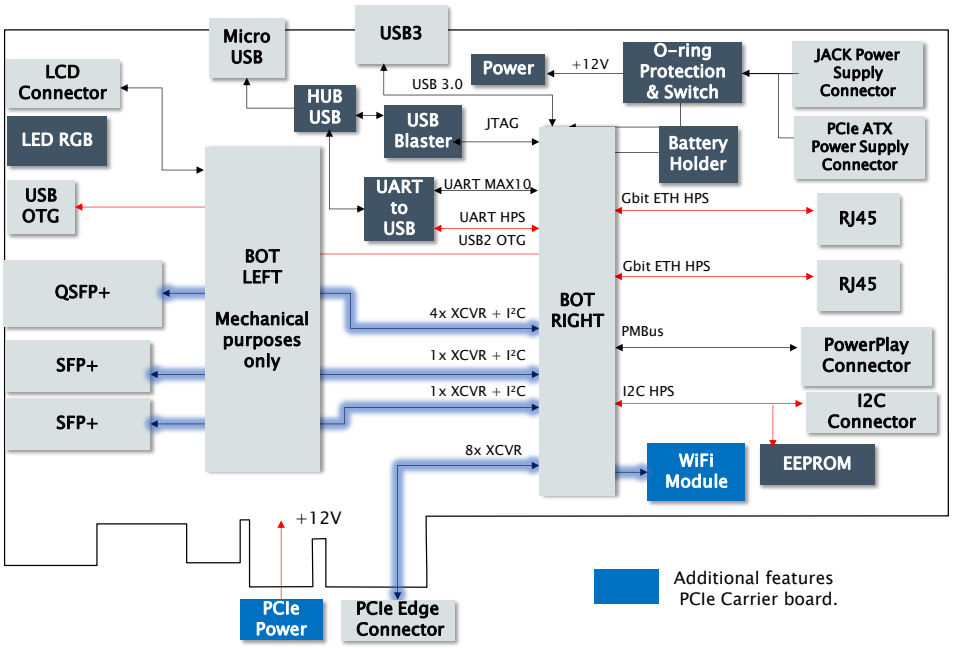



*Figure 9: PCIe Carrier Board Block Diagram*

## 3.2 Board Layout

The Arria® 10 SoC module is based on Altera Arria® 10SX SoC FPGA in the FBGA F34 package:

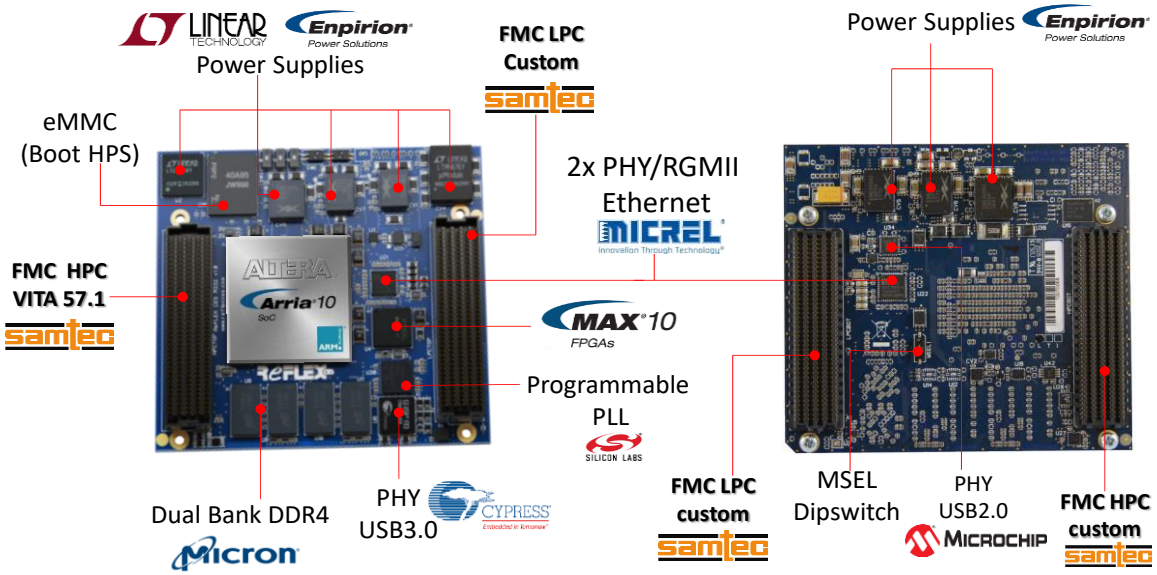
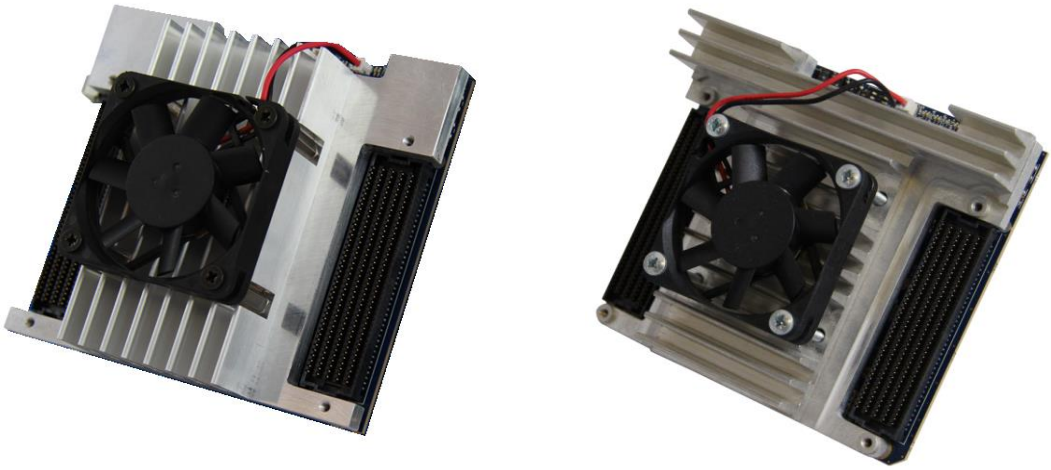



*Figure 10: Layout and Components*



SoM A10 SoC Module V1.0 SoM A10 SoC Module V2.0

*Figure 11: SoM A10 SoC module with air cooled system*

*NOTE: A specific chapter is dedicated for SoM A10 SoC Lite version refer to 3.3. to see its specificities.*

### 3.2.1 Achilles Starter Board

The following figure shows the features of the Achilles Starter Board. The board size is 100mm (3.94 in) x 140mm (5.51 in):

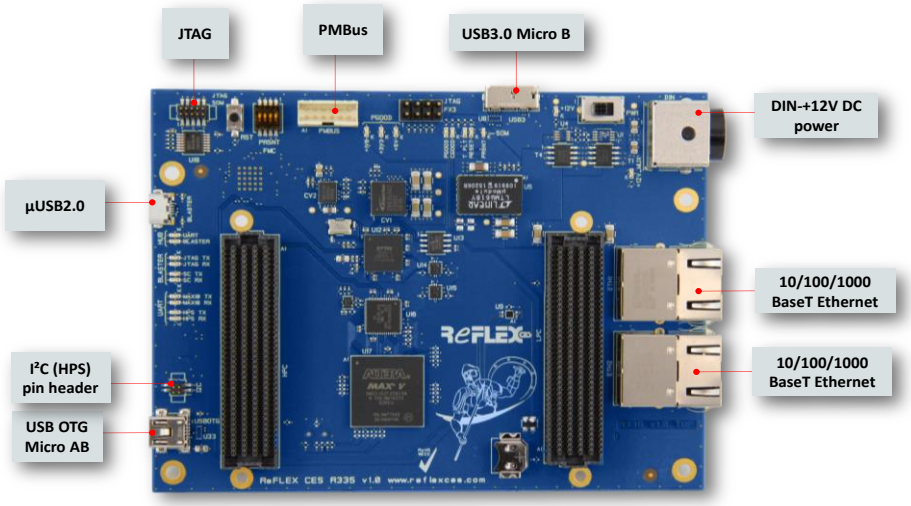



*Figure 12: Achilles Starter Board*

### 3.2.2 PCIe Carrier Board

The following figure shows the features of the PCIe Carrier Board. The board size is 111.5mm (4.376 in) x 167.65mm (6.6 in).

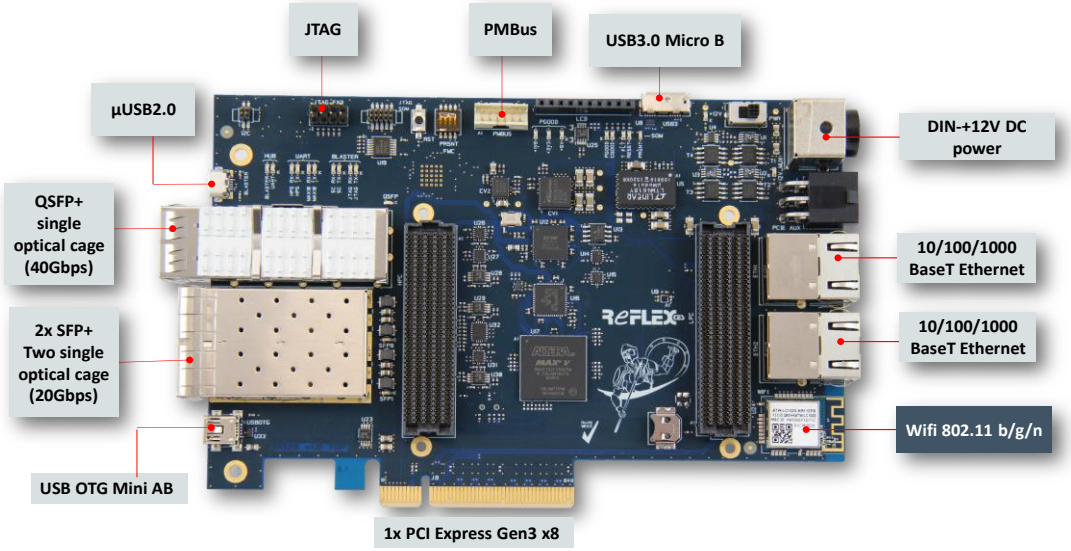



*Figure 13: PCIe Carrier Board*

### 3.2.3 Feature List and Details

The Arria® 10 SoC SoM features a highly integrated Altera Arria® 10 SoC module and is intended to be used with application-specific baseboards via industry-standard FMC connectors. This board is designed to have the smallest possible form factor, while using the maximum capacities of the Arria® 10 SoC.

The following list describes the features of the Arria® 10 SoC SoM:

| Feature | Description |
|---|---|
| Arria® 10 SoC FPGA 10AS066H2F34I1HG. | • Dual ARM® Cortex™-A9 MPCore™ processor-based hard processor system (HPS)<br>• Speed grade-1<br>• 660 KLE - F34 package (1152 pins) - Industrial temperature grade |
| 2x DDR4 SDRAM up to 2400MT/s | For Indus version:<br>• 32-bit wide bank for FPGA (2GB)<br>• 32-bit wide bank for HPS (2GB)<br>For Turbo version:<br>• 32-bit wide bank for FPGA (4GB)<br>• 32-bit wide bank for HPS (4GB) |
| System-on-Module usage (Bottom connectors only) | • 1x FMC LPC bottom-right connector with HPS signals and 14 transceivers<br>• 1x FMC HPC or LPC bottom-left connector depending on FPGA density<br>○ LPC bottom connector for 270 KLE density with 10 transceivers<br>○ HPC bottom connector for 660 KLE density with 10 transceivers |
| Carrier board usage (Bottom and Top connectors) | In addition to the System-on-Module usage:<br>• 1x FMC LPC top-right connector with 10 transceivers, 1.8V VITA 57.1<br>• 1x FMC top-left connector with 10 transceivers, 1.8V VITA 57.1<br>○ LPC Top connector for 270 KLE density<br>○ HPC Top connector for 660 KLE density |
| LVDS, LVCMOS & XCVR | • 24 transceivers up to 11.3Gbps<br>• 80 LVDS bidirectional (or 160 single-ended) on FMC HPC Top connector<br>• 34 LVDS bidirectional (or 64 single-ended) on FMC LPC Top connector |
| USB 3.0 Device | • USB 3.0 device connectivity using Cypress FX3 super speed controller (660 KLE version) |
| USB 2.0 Device/Host | • USB OTG support connected to the HPS |
| System Controller | • MAX® 10 CPLD for system control |
| 2x Gigabit Ethernet | • 10/100/1000 Base-T Ethernet interfaces connected to the HPS |
| Serial interfaces | • I²C<br>• UART connected to the HPS |




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

| Feature | Description |
|---|---|
| | • UART connected to the System Controller |
| Soft programming /eMMC | • Store U-boot, Kernel and RootFS (32GB on 660 KLE), HPS programming |
| FPGA programming | • AS configuration supported with EPCQL512, remote upgrade and failsafe |
| Power Management | • PMBus support<br>• Power consumption: 35W typ., 55W max (without FMC), 103W max (with Dual FMCs) |

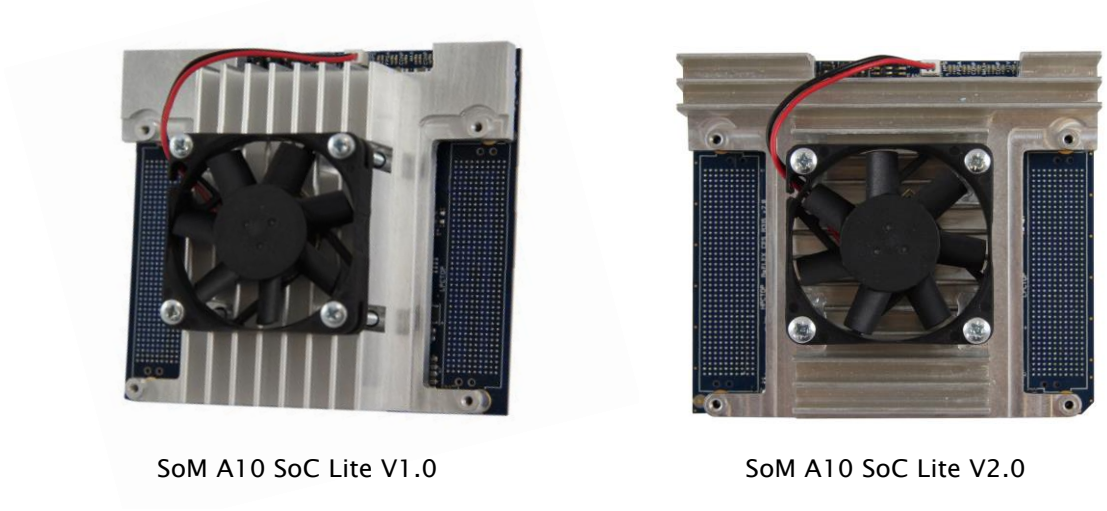
*Table 2: Board features list and details*

## 3.3 SoM A10 SoC Lite Version

### 3.3.1 Overview

The SoM A10 SoC Lite version offers similar functions and services as the SoM A10 module Turbo/Indus versions except for some components not present on the board or which have different characteristics (refer to the table below to see the board lite features).

The SoM A10 SoC Lite version can be assembled with the Achilles Starter board or the PCIe Carrier board.



SoM A10 SoC Lite V1.0 SoM A10 SoC Lite V2.0

*Figure 14: SoM A10 SoC Lite version with air cooled system*

### 3.3.2 Feature List and Details

The SoM A10 SoC Lite version is delivered with these specificities:

| Feature | Description |
|---|---|
| Arria® 10 SoC FPGA<br>**10AS027H3F34E2SG** | • Dual ARM® Cortex™-A9 MPCore™ processor-based hard processor system (HPS)<br>• **Speed grade -2**<br>• 270 KLE - F34 package (1152 pins) - Commercial temperature grade |
| 2x DDR4 SDRAM<br>up to **2133 MT/s** | • 32-bit wide bank for FPGA _ 2GB with 270 KLE version<br>• 32-bit wide bank for HPS _ 2GB with 270 KLE version |
| System-on-Module usage (Bottom connectors only) | Compliance ANSI VITA 57.1 FMC, Mezzanine & Carrier board, but FMC LPC/HPC connectors are not connected on Top side of the module board.<br>• 1x VITA57.1 FMC HPC<br>• 1x FMC LPC-RXC+ 9x XCVRs channel<br>• LVCMOS 1.8V compliant Only, BOTTOM interconnect only. |
| Carrier board usage (Bottom and Top connectors) | In addition to the System-on-Module usage:<br>• 1x FMC LPC top-right connector with 10 transceivers, 1.8V VITA 57.1<br>• 1x FMC top-left connector with 10 transceivers, 1.8V VITA 57.1<br>  ○ LPC Top connector for 270 KLE density<br>  ○ HPC Top connector for 660 KLE density |
| USB 2.0 Device | • USB OTG support connected to the HPS |
| System Controller | • MAX® 10 CPLD for system control |
| 2x Gigabit Ethernet | • 10/100/1000 Base-T Ethernet interfaces connected to the HPS |
| Serial interfaces | • I²C<br>• UART connected to the HPS<br>• UART connected to the System Controller |
| Soft programming /eMMC | • Store U-boot, Kernel and RootFS (**8GB** on 270 KLE), HPS programming |
| FPGA programming | • AS configuration supported with EPCQL512, remote upgrade and failsafe |
| Power Management | • PMBus support |

*Table 3: Board features list and details for SoM A10 SoC Lite version*

# 4. PRODUCT SPECIFICATIONS

## 4.1 Mechanical Description

The following diagram illustrates the mechanical dimension of the Achilles board without the heatsink and FMC daughter card. All measurements shown are for information only. Please contact REFLEX support (see section 1.2) for 3D files.

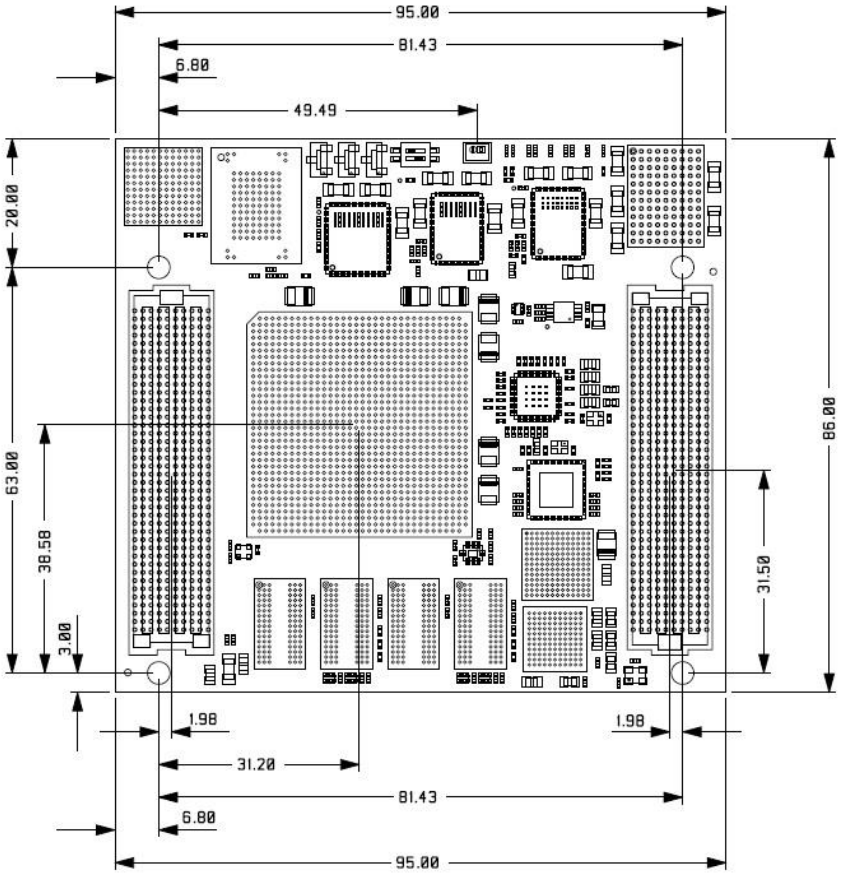



*Figure 15: Module Mechanical Dimensions*

The following diagram illustrates the mechanical dimensions of the Achilles Starter Board:

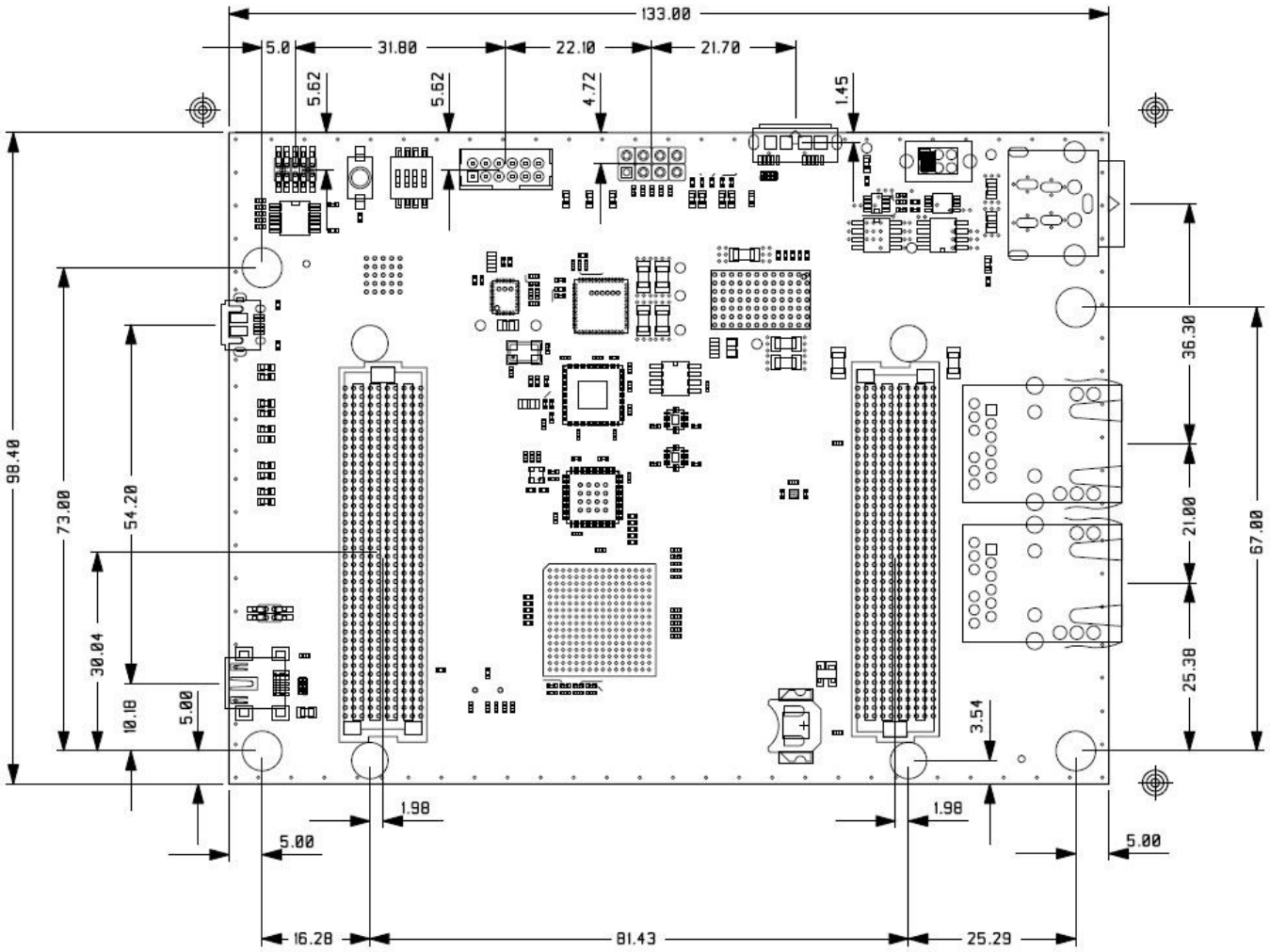



*Figure 16: Achilles Starter Board Mechanical Dimensions*

*NOTE: For PCIe Carrier Board dimensions, refer to Reference Manual of the PCIe Carrier Board. For more details, please contact REFLEX support at [https://support.reflexces.com/.](https://support.reflexces.com/)*

## 4.2 Electrical Specification

### 4.2.1 Power Input

Achilles DevKit boards can be supplied from several independent power inputs depending of the boards use. Power supplies needed when the boards are installed inside a server or a computer come from PCIe Carrier board:

- PCIe edge finger (only 25W, as required by PCI Express CEM Specification Rev3.0) and,
- Auxiliary Power connector

Power supply needed when the boards are used on table comes from PCIe Carrier board:

- DIN +12V DC power

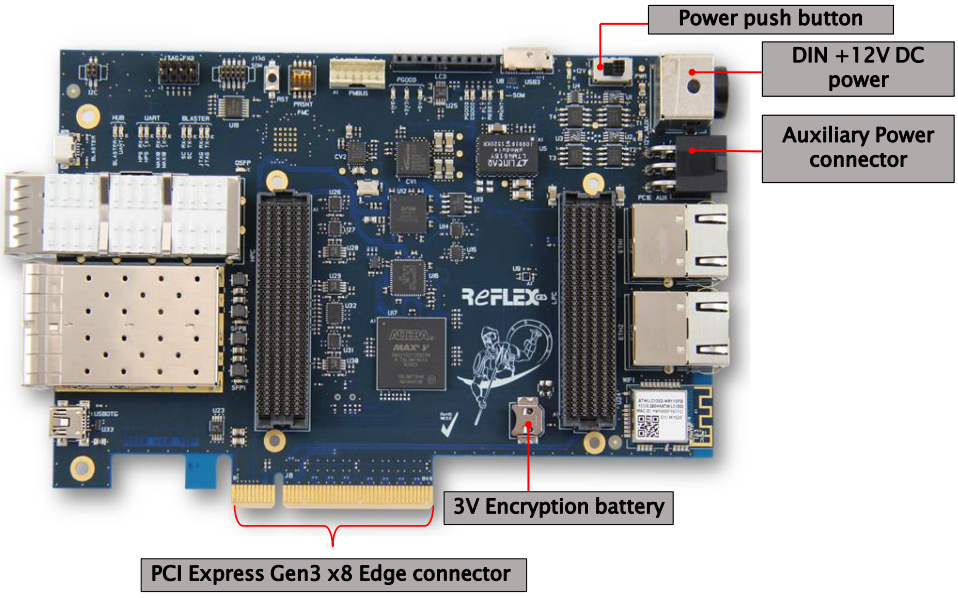
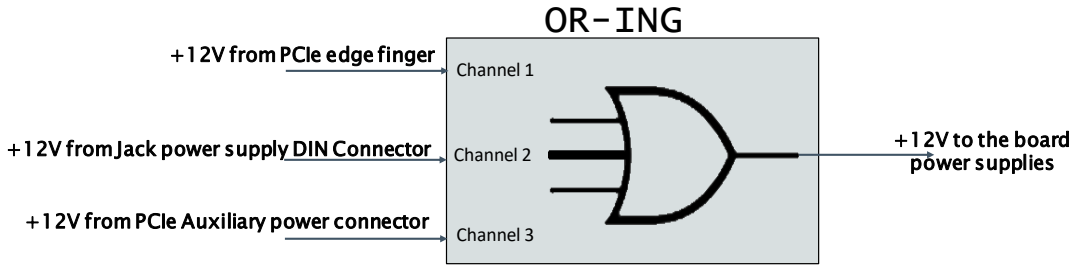



*Figure 17: PCIe Carrier board power supplies*

The PCIe Carrier board integrate an OR-ing function that allows the following features:

- Hot-Swap
- Inrush current limiting
- Overcurrent protection
- Maximum current setting on both channel




The voltage range is compliant with PCIe Specification Rev 3.0: +12V (+/-8%).




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

### 4.2.2 Voltage Rails

The +12V rail generates most of the power supplies of the board (+12V, +5V, +3,3V +2,5V +1,8V +1,2V +0,95V +0,9V +0,6V). The +2,5V is generated from the +3,3V rail.

The table below shows the different power supplies of Achilles DevKit boards and the associated components:

| | +0,6 | +0,9 | +0,95 | +1,2 | +1,8 | +2,5 | +3,3 | +5V | +12V |
|---|---|---|---|---|---|---|---|---|---|
| FPGA | | X | X | X | X | | | | |
| DDR4 | | | | X | | | | | |
| VTT DDR4 | X | | | | | X | | | |
| MAX® 10 | | | | | | | X | | |
| eMMC | | | | | | | X | | |
| PHY GbE | | | | X | | | X | | |
| PHY USB OTG | | | | | | | X | | |
| OSCILLATOR | | | | | X | | | | |
| PLL | | | | | X | | | | |
| FMC LPC | | | | | | | X | | X |
| FMC HPC | | | | | | | X | | X |
| DC/DC _ ATX input | | | | | | | | X | X |
| FX3 | | | | X | | | | | |

*Table 4: Power supplies of Achilles DevKit boards*

### 4.2.3 Power Consumption

The amount of the power consumed by the ACHILLES DevKit boards is highly dependent on the use case (number of FPGA resources, memory, transceiver, etc…), and the environment (ambient temperature, airflow).

Power consumption of the board is the following:

- 35W typ.,
- 55W max (without FMC),
- 103W max (with Dual FMCs)

In order to dissipate the Arria® 10 FPGA to the maximum, a VHDL "Stress" design is presented below. The stress test has taken place with the following configuration for the SoM A10 SoC Turbo version:

- Environment:
  - Chamber reference: BINDER MK53
  - Chamber temperature order: 25°C
  - Chamber fan speed: 100%

- Board:
  - Board interfaces:
    - 10 XCVR FMC HPC 5.0Gbps

- 10 XCVR FMC LPC 5.0Gbps
- 40 LVDS FMC HPC pairs 1.0Gbps
- 16 LVDS FMC LPC pairs 1.0Gbps
- DDR4 component 2400Mbps

- Stresser design:
  - Stresser logic frequency: 250MHz
  - Stresser Block RAM frequency: 250MHz
  - Stresser Block DSP frequency: 250MHz
  - Timing models: Final
  - Logic utilization (in ALMs): 247,677 / 251,680 (98%)
  - Total registers: 490088
  - Totals pins: 561 / 604 (93%)
  - Total block memory bits: 42,591,616 / 43,642,880 (98%)
  - Total RAM Blocks: 2,127 / 2,131 (100%)
  - Total DSP Blocks: 1,600 / 1,687 (95%)
  - Total HSSI RX channels: 20 / 24 (83%)
  - Total HSSI TX channels: 20 / 24 (83%)
  - Total PLLs: 36 / 64 (56%)

| Conditions | | Results | |
|---|---|---|---|
| Chamber fan (%) | Toggle Rate (%) | FPGA Die Temperature (°C) | Board Power Consumption (W) |
| 100 | 0 | 54.0 | 35.76 |
| 100 | 10 | 59.6 | 39.72 |
| 100 | 15 | 61.6 | 41.88 |
| 100 | 20 | 64.4 | 44.76 |
| 100 | 25 | 67.2 | 47.04 |
| 100 | 30 | 69.2 | 49.68 |
| 100 | 35 | 72.0 | 53.04 |
| 100 | 40 | 79.6 | 58.44 |
| 100 | 45 | 81.7 * | 63.00 |

*Table 5: Power consumption for SoM A10 SoC Turbo version*

*: With a toggle rate of 45% we reach the limit of the core alimentation of the module. The core alimentation shut down after a few minutes of stress with a toggle rate of 45%.

*NOTE: Results for SoM A10 SoC Indus version can be assimilated to SoM A10 SoC Turbo version as the maximum power consumption of the board. DDR4 memories being less important for SoM A10 SoC Indus version, it can be considered that the power consumption of the board is slightly lower.*

The stress test has taken place with the following configuration for the SoM A10 Lite version:

- Environment:
  - Chamber reference: BINDER MK53
  - Chamber temperature order: 25°C
  - Chamber fan speed: 100%

- Board:
  - Board interfaces:
    - 10 XCVR FMC HPC 10.3125Gbps
    - 10 XCVR FMC LPC 10.3125Gbps
    - 17 LVDS FMC HPC pairs 1.0Gbps
    - 16 LVDS FMC LPC pairs 1.0Gbps
    - DDR4 component 2133Mbps

  - Stresser design:
    - Stresser logic frequency: 250MHz
    - Stresser Block RAM frequency: 250MHz
    - Stresser Block DSP frequency: 250MHz
    - Timing models: Final
    - Logic utilization (in ALMs): 93,176 / 101,620 (92%)
    - Total registers: 159140
    - Totals pins: 478 / 496 (96%)
    - Total block memory bits: 13,738,880 / 15,360,000 (89%)
    - Total RAM Blocks: 717 / 750 (96%)
    - Total DSP Blocks: 768 / 830 (93%)
    - Total HSSI RX channels: 24 / 24 (100%)
    - Total HSSI TX channels: 24 / 24 (100%)
    - Total PLLs: 35 / 48 (73%)

| Conditions | | Results | |
|---|---|---|---|
| Chamber fan (%) | Toggle Rate (%) | FPGA Die Temperature (°C) | Board Power Consumption (W) |
| 100 | 0 | 56.1 | 33.24 |
| 100 | 10 | 58.2 | 34.08 |
| 100 | 15 | 58.9 | 34.44 |
| 100 | 20 | 59.6 | 34.80 |
| 100 | 25 | 60.3 | 35.28 |
| 100 | 30 | 60.3 | 35.64 |
| 100 | 35 | 60.9 | 36.12 |
| 100 | 40 | 60.9 | 36.48 |
| 100 | 45 | 61.6 | 36.96 |
| 100 | 50 | 63.0 | 37.32 |
| 100 | 55 | 63.0 | 37.80 |
| 100 | 60 | 63.7 | 38.16 |
| 100 | 65 | 63.7 | 38.64 |
| 100 | 70 | 64.4 | 39.12 |
| 100 | 75 | 65.1 | 39.48 |
| 100 | 80 | 65.8 | 39.96 |
| 100 | 85 | 66.5 | 40.44 |
| 100 | 90 | 67.2 | 40.92 |
| 100 | 95 | 67.9 | 41.40 |
| 100 | 100 | 67.9 | 41.88 |

*Table 6: Power consumption for SoM A10 SoC Lite version*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 4.3 Thermal Specifications

A heatspreader, a heatsink, and a fan are supplied with and mounted on every SOM and Instant DevKit.


You can choose a conduction-cooled or air-cooled solution, depending on your environment, as shown in the figures below:

## 4.4 Regulatory Compliance

| | |
|---|---|
| Overview | This product is designed and tested to conform to the European Union Directives and standards described in this section. |
| Declaration of conformity | For technical support questions, please first contact REFLEX support at https://support.reflexces.com/ to create an account on the REFLEXCES.com website. |
| Directives | 2011/65/EU on the restriction hazardous substances (RoHS)<br>2014/30/EU Directive Electromagnetic Compatibility (EMC)<br>2014/35/EU Low Voltage Directive (LVD) |

### 4.4.1 Electromagnetic Compatibility

EN 55022:2010 Information Technology Equipment Radio Disturbance Characteristics _ Limits and Methods of measurement.

EN 55024:2010 Information Technology Equipment Immunity Characteristics _ Limits and Methods of measurement.

This is a Class A product. In a domestic environment, this product can cause radio interference, in which case the user might be required to take adequate measures.

*NOTE* *For reliability questions, please contact REFLEX quality service at certification@reflexces.com.*

### 4.4.2 Safety

a) Markings

This product complies with Directive 2012/19/EU on waste electrical and electronic equipment (WEEE). The affixed product label indicates that the user must not discard this electrical or electronic product in domestic household waste.

CE

This product complies with CE Directives 2011/65/EU on the restriction hazardous substances (RoHS) in electrical and electronic equipment, 2014/35/EU Low Voltage Directive (LVD), 2014/30/EU Directive Electromagnetic Compatibility (EMC).

b) Certification

- EU RoHS: Certificate of Compliance
- REACH: Certificate of Compliance
- Conflict Minerals CMRT: 5.0

## 4.5 Reliability

For reliability questions, please contact REFLEX quality service at certification@reflexces.com.

# 5. BOARD INSTALLATION

**WARNING:** As for all electronic devices, it is recommended to use ESD protection equipment when handling the board. We strongly suggest following standard ESD care when touching any part of the board. (Refer to section 1.5 "Electrostatic Discharge").

Do not use the board in a wet place or with wet hands.

**CAUTION** Electrostatic discharge (ESD) can damaged the board. Follow standard ESD prevention measures when handling the board.

To avoid damaging the board and your security, please contact REFLEX support at https://support.reflexces.com/ for all defective elements/components/accessories needed to change.

The following figure provides an overview of the different connector's topology, when both the FPGA mezzanine card and the carrier board are mounted:

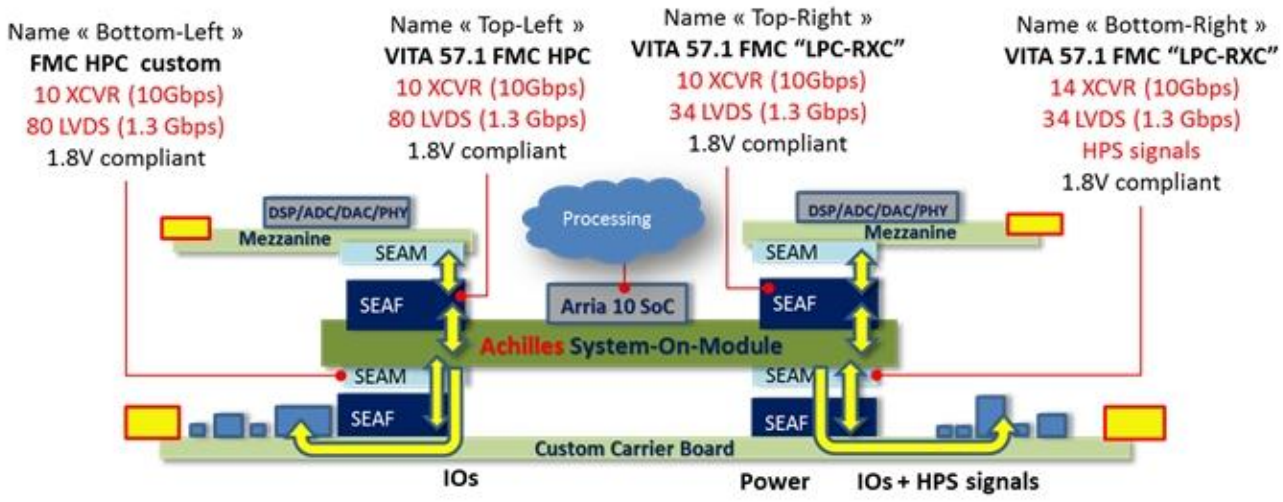



*Figure 18: Connector view*

The following figure shows several different use cases for the Arria® 10 SoC SoM:

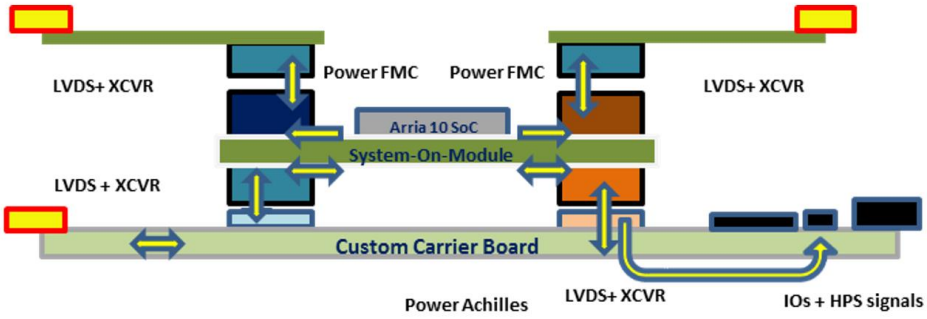



*Figure 19: Use Case 1*

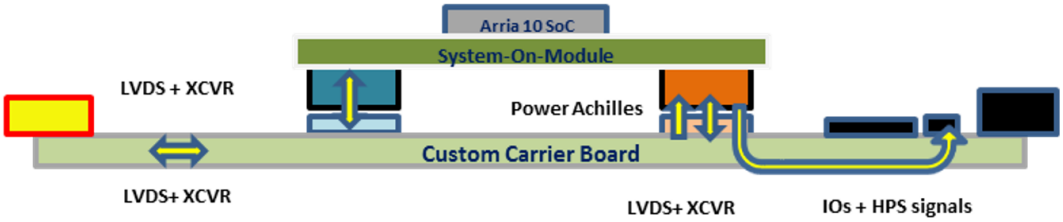



*Figure 20: Use Case 2*

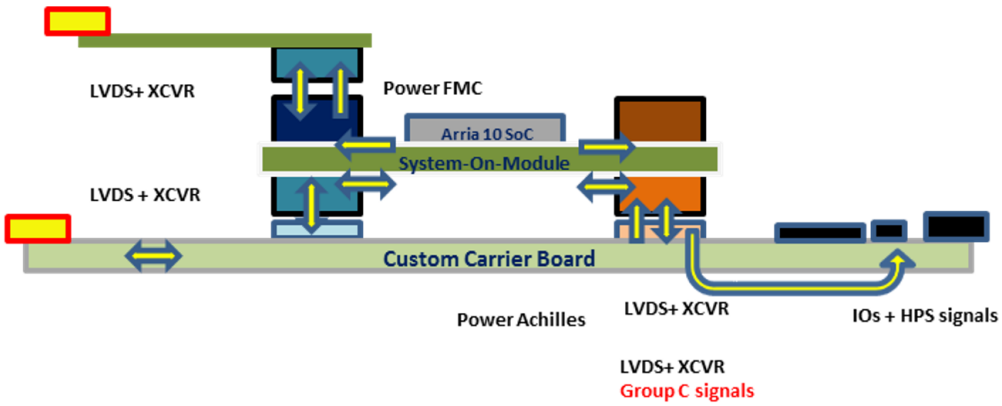



*Figure 21: Use Case 3*

# 6. BOARD CONFIGURATION

## 6.1 Tools Configuration

You can use the latest version of Quartus II to configure the Arria® 10 and the MAX® 10. There is no need for an external USB Blaster probe, as one is embedded on the Starter Board.

Use the USB standard A-type cable provided and plug it in to the micro USB connector.

## 6.2 Innovative Graphical User Interface (GUI)

The Graphical User Interface provided with the Achilles DevKit (Turbo and Indus board versions) demonstrates the performance of the SoM module using several menu interfaces.

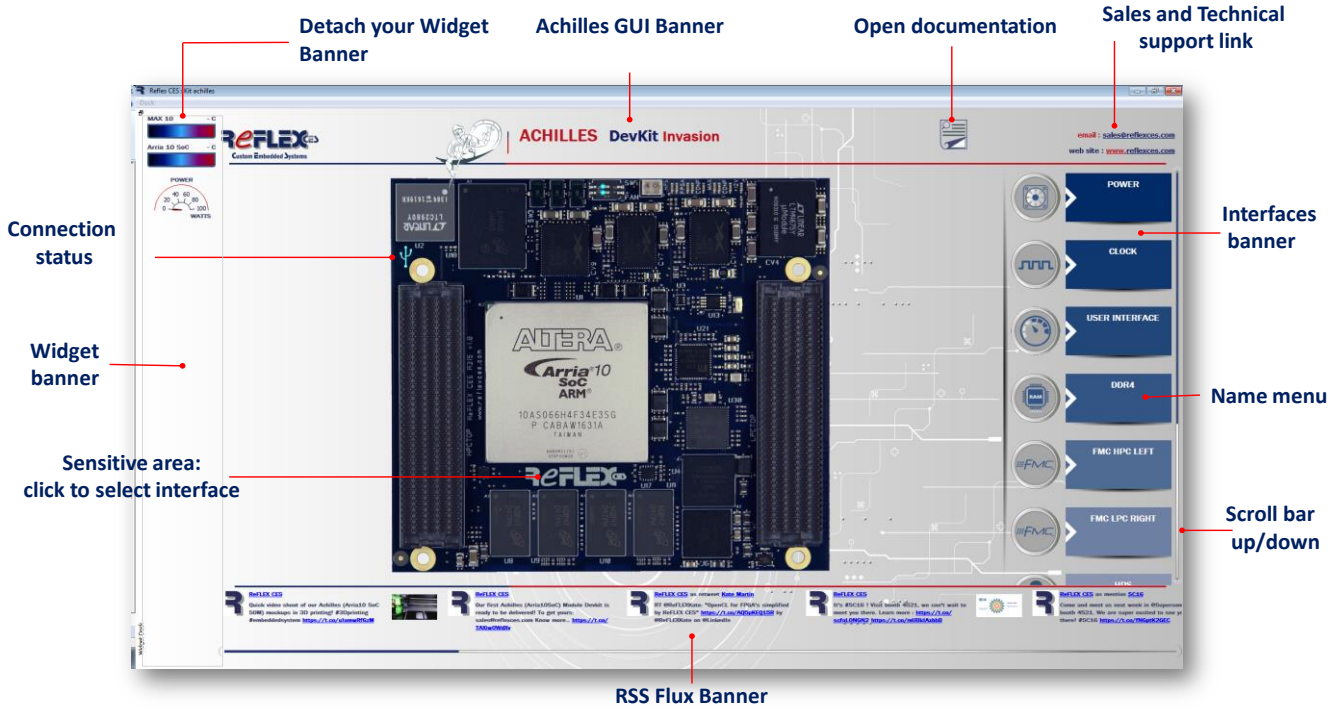



*Figure 22: Board GUI*

To run the GUI, click the launcher file. The main panel shown above appears, enabling you to access information about the board and its various interfaces, run test designs, and understand the HDL design. The GUI also enables you to access the board's documentation and technical support.

## 6.3 Board Programming Methods

You can program the board using either JTAG or AS (Active Serial) programming. Both these methods are described below.

- **JTAG programming**: Using this method of programming, named after the IEEE standards Joint Test Action Group, the configuration bitstream is downloaded directly into the FPGA. The FPGA retains this configuration as long as power is supplied to the board; the configuration information is lost when the power is turned off.
- **AS programming**: The configuration bitstream is downloaded into the Altera EPCQL-512 serial configuration device. This method provides non-volatile storage of the bitstream, so that the information is retained even when the power supply to the Arria® 10 DevKit is turned off. When the board power is turned on, the configuration data in the EPCQL-512 device is automatically loaded into the Arria® 10 FPGA.

  **WARNING:** In ASx4 mode, if using transceivers during the Arria® 10 FPGA start-up from the EPCQ-L512, it is possible that the transceivers reference clocks are not running and stable, especially if the reference clock is provided by the Si5341. Consequently, you need a recalibration of the transceivers. Refer to the chapter "User Recalibration" (p578) from the Intel® Arria® 10 Transceiver PHY User Guide: https://www.intel.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/arria-10/ug_arria10_xcvr_phy.pdf

The figure below illustrates both these programming methods:

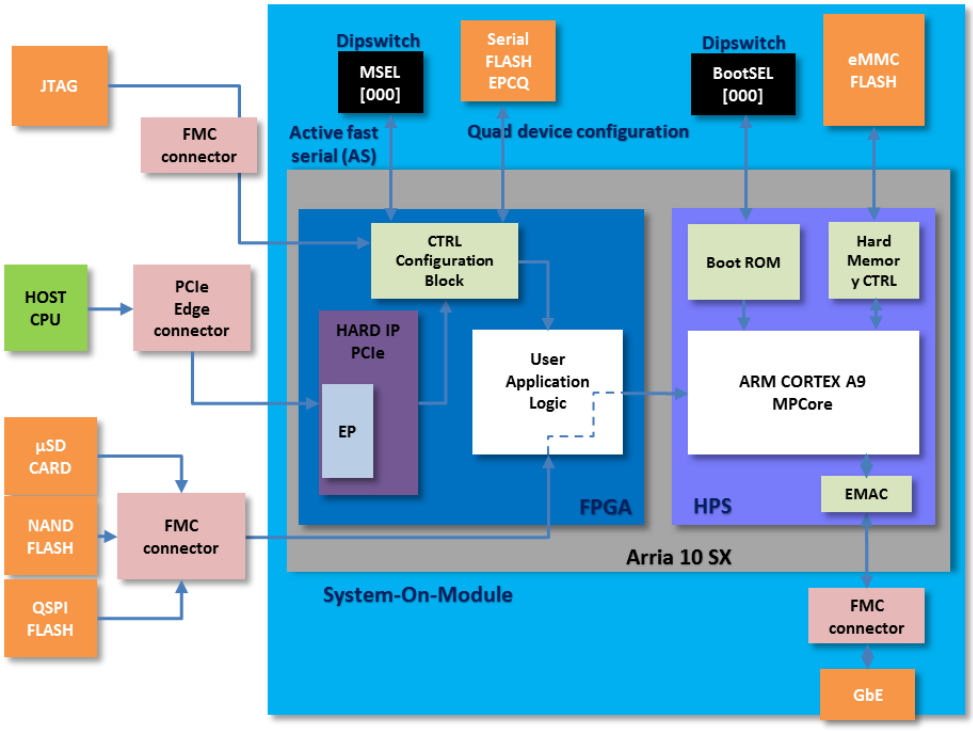



*Figure 23: Board Programming*

- **FPP x32**: The configuration bitstream is downloaded from the HPS. This method requires the HPS to be configured to download the FPGA bitstream into the FPGA. At each boot, the configuration must be download from the HPS, this requires the HPS to store the bitstream either on a network device or into its storage space.

In the Arria® 10 SoC, several boot configurations are possible:

- The HPS boot and FPGA configuration occur separately (Achilles DevKit configuration)
- The HPS boots first and configures the FPGA
- The HPS boots from the FPGA after the FPGA is configured

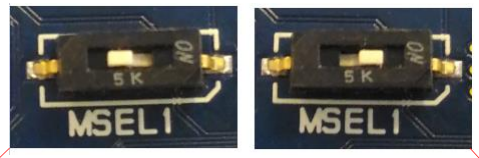
This means that there are several options for board configuration:

- You can program the Arria® 10 FPGA and (or) the MAX® 10 through the USB blaster in JTAG mode.

- At power-up, the Arria® 10 FPGA automatically loads via a quad serial interface with a dedicated FLASH EPCQ in AS configuration mode.

- After having initialized the HPS, the BootROM determines the device where the bootloader is stored using the BSEL pins (NAND, QSPI, µSD…the FPGA IOs can be used).

- After having initialized and readied the HPS for use, you can load a bitstream from an external source using GbE to program the FPGA (in order to use this mode, the MSEL dipswitch under the board must be in FPP mode). Below are the MSEL1 dipswitch positions:




**OFF** : Configuration mode « ASx4 ». **ON**: Configuration mode « FPPx32 ».

- "Configuration via Protocol": a minimal configuration is contained in the EPCQ. This binary is loaded in the FPGA, enabling improved PCIe access and allowing the software to load an external image in the Arria® 10. See Section 7.8 for more information about CvP.

For more information about board configuration, see Chapter 8: HPS.

# 7. HARDWARE DESCRIPTION

## 7.1 Signal Pinouts General Description

| Type | Description |
|---|---|
| NC | Not Connected |
| I | Input |
| O | Output |
| I/O-3V3 | Input/Output 3.3V |
| I-3V3 | Input 3.3V |
| O-3V3 | Output 3.3V |
| I/O-1V8 | Input/Output 1.8V |
| I-1V8 | Input 1.8V |
| O-1V8 | Output 1.8V |
| DP-I/O | Differential Pair Input/Output |
| XCVR-I | Transceiver Input |
| XCVR-O | Transceiver Output |
| DP-I | Differential Pair Input |
| DP-O | Differential Pair Output |
| PU | Pull-up Resistor |
| PD | Pull-down Resistor |
| PWR | Power connection |

*Table 1: Signal Descriptions*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 7.2 Clocks Description

### 7.2.1 Clock Tree

The picture below shows connection of dedicated clocks.

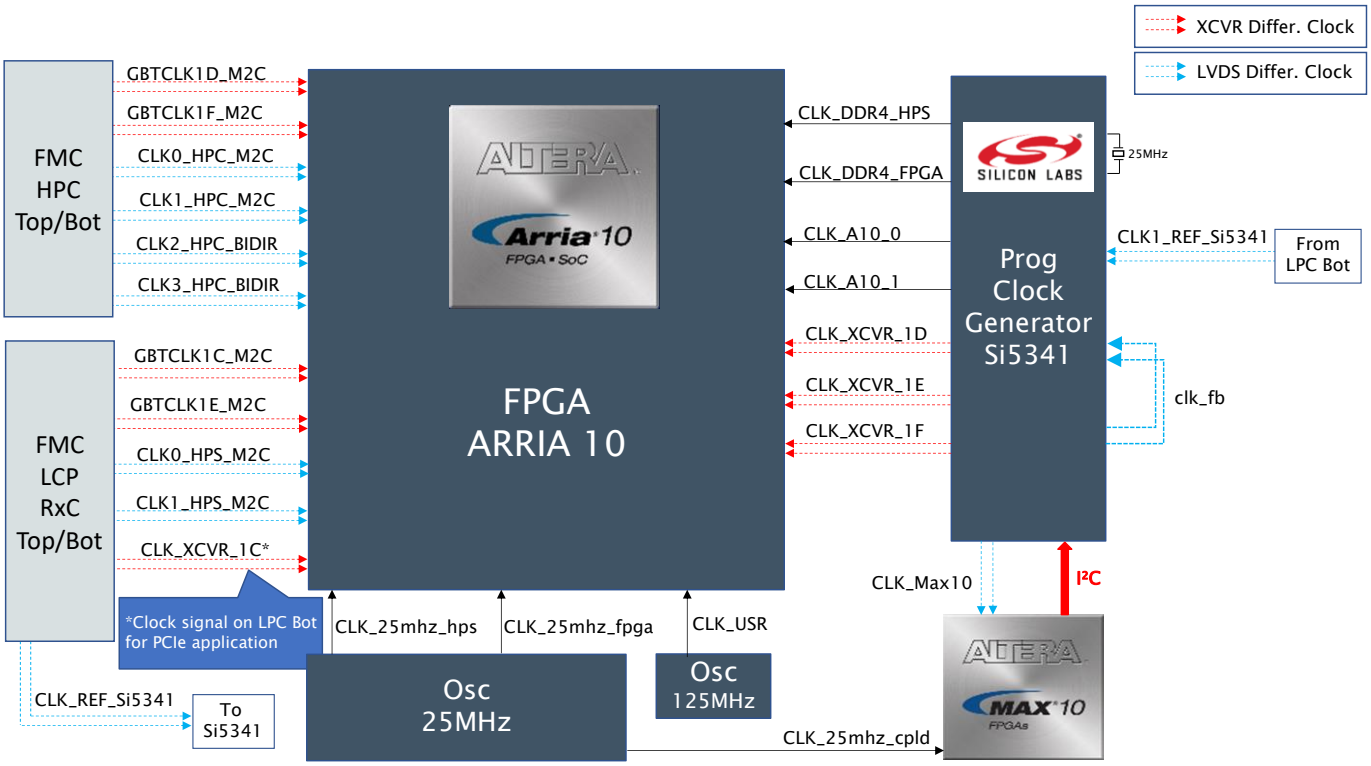



*Figure 24: Clock Tree Overview*

### 7.2.2 Clocks Assignments

The table below shows pin assignment for the clock generator:

| SIGNAL CLOCK | DESCRIPTION | COMPONENT PIN LABEL | IO STANDARD | CONNECTED TO |
|---|---|---|---|---|
| clk_25mhz_fpga | 25MHz clock input | IO24/LVDS3A_12_p/CLK1_p/DQ61(3A) | LVCMOS 1V2 | Arria® 10 Pin AJ9 |
| clk_25mhz_hps | 25MHz clock input | HPS_CLK1 | LVCMOS 1V8 | Arria® 10 Pin B16 |
| clk_25mhz_cpld | 25MHz clock input | IO/CLK0_n/DIFFIO_RX_L20_n(2) | LVCMOS 1V8 | MAX® 10 Pin G5 |
| clk_usr | 125MHz clock input | IO13/LVDS2A_18_n/CLKUSR/DQ30(2A) | LVCMOS 1V8 | Arria® 10 Pin AK16 |
| usbotg_clk | 26MHz clock input | IO12/ALE/USB1_CLK/EMAC0_TX_CLK/DQ2 | LVCMOS 1V8 | Arria® 10 Pin J20 |

*Table 2: Pin assignments for the clocks from oscillators*

| SIGNAL CLOCK | DESCRIPTION | COMPONENT PIN LABEL | IO STANDARD | CONNECTED TO |
|---|---|---|---|---|
| clk_max10_n | Clock output #0 from external PLL SI5341 | IO/CLK1_n/DIFFIO_RX_L22_n (2) | LVDS | MAX® 10 Pin H5 |
| clk_max10_p | | IO/CLK1_p/DIFFIO_RX_L22_p (2) | | MAX® 10 Pin H4 |
| clk_A10_0 | Clock output #1 from external PLL SI5341 | IO24/LVDS2K_12_p/CLK1_p/DQ 5 (2K) | LVCMOS 1V2 | Arria® 10 Pin E23 |
| clk_A10_1 | Clock output #2 from external PLL SI5341 | IO34/LVDS3A_7_p/DQ61 (3A) | LVCMOS 1V2 | Arria® 10 Pin AH8 |
| clk_xcvr_1d_n | Clock output #4 from external PLL SI5341 | REFCLK_CHT_n (1D) | LVDS | Arria® 10 Pin Y27 |
| clk_xcvr_1d_p | | REFCLK_CHT_p (1D) | | Arria® 10 Pin Y28 |
| clk_xcvr_1e_n | Clock output #5 from external PLL SI5341 | REFCLK_CHB_n (1E) | LVDS | Arria® 10 Pin V27 |
| clk_xcvr_1e_p | | REFCLK_CHB_p (1E) | | Arria® 10 Pin V28 |
| clk_xcvr_1f_n | Clock output #6 from external PLL SI5341 | REFCLK_CHT_n (1F) | LVDS | Arria® 10 Pin M27 |
| clk_xcvr_1f_p | | REFCLK_CHT_p (1F) | | Arria® 10 Pin M28 |
| clk_ddr4_hps | Clock output #7 from external PLL SI5341 | IO24/LVDS2J_12_p/CLK1_p/DQ 9 (2J) | LVCMOS 1V2 | Arria® 10 Pin AL27 |
| clk_ddr4_fpga | Clock output #8 from external PLL SI5341 | IO22/LVDS3A_13_p/CLK0_p/DQ 62 (3A) | LVCMOS 1V2 | Arria® 10 Pin AL5 |
| clk_fb_p | Clock output #9 from external PLL SI5341 | FB_IN | LVDS | Si5341 pin 61 |
| clk_fb_n | | FB_IN | | Si5341 pin 62 |

*Table 3: PLL On Board Clock Output*

**Note: For the pinout assignment of clocks from the HPC and LPC connectors please refer to the Superpinout document.**

### 7.2.3 Programmable Clock Generator

The Arria® 10 SoC SoM includes three clocks of different frequencies, as well as a programmable clock generator.

- A 25 MHz clock is used for the FPGA, the HPS, and the MAX® 10 CPLD.
- A 26 MHz clock is used for the USB 2.0 OTG PHY.
- A 125 MHz clock is used for FPGA configuration.

The programmable clock generator is an SI5341 device from Silicon Labs that can generate up to 10 any-frequency output clocks and provides high quality clock signals for high-speed transceivers.

The clock generator can be programmed in-circuit by the MAX® 10 through the I²C serial interface. You can modify the frequency of any generated clocks between 0.0001 MHz to 750 MHz.

See the SI5341 device datasheet from Silicon Labs for more information:
https://www.silabs.com/Support%20Documents/TechnicalDocs/Si5341-40.pdf

## 7.3 Power Management

The following figure illustrates the power distribution for the Arria® 10 SoC SoM:

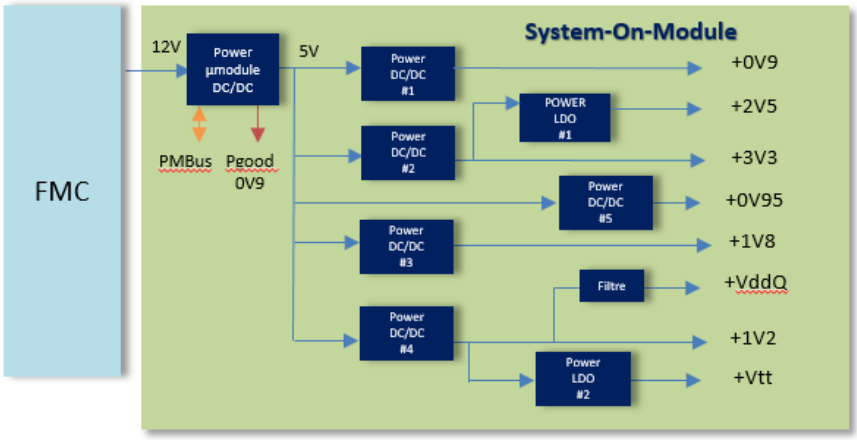



*Figure 25: Power Distribution*

The Power System Manager on board enables you to:

- Power On/Off each power rail independently
- Dynamically monitor the current and supply for each power rail
- Manage the power sequencing by programming the supervisor on board

The PMBus is easy to connect with the USB cable DC1613A (available on the Starter Board or PCIe carrier board). The following figure illustrates the power signals to the supervisor:

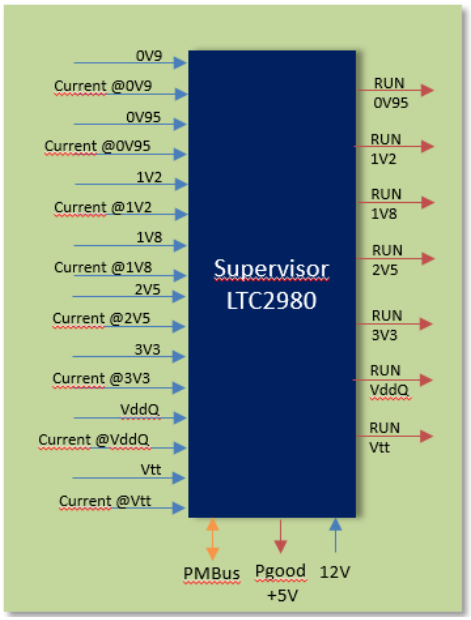



*Figure 26: Power Signals to Supervisor*

## 7.4 SoM Connectors

Please refer to the Superpinout document.

## 7.5 Board LEDs


*Figure 27: LED positions*

### 7.5.1 Status LEDs

The FPGA development board includes board-specific status LEDs to indicate board status. The table below describes the LED indicators:

| Board Reference | Signal Name | Description | Pin |
|---|---|---|---|
| DS12 | +12V Power Supply | On green when +12V power is active. | -- |

*Table 4: Power LEDs status*

| Board Reference | Signal Name | Description | Connected to |
|---|---|---|---|
| DS7 | max10_confdone | On Red when MAX® 10 FPGA is not programming and off when programming is done. | MAX® 10 pin C5 |
| DS2 | fpga_conf_done | On Red when Arria® 10 FPGA is not programming and off when programming is done. | FPGA pin AM12 |

*Table 5: FPGA LEDs status*

### 7.5.2 User LEDs

There are two user-controllable LEDs on the board directly connected to the Arria® 10 FPGA and two LEDs connected to the HPS of the FPGA. There are also two user-controllable LEDs connected to the MAX® 10 device.

These LEDs can be lit green and red. Each LED is driven directly by a pin on the Arria® 10 FPGA. Driving the associated pin to a low logic level turns the LED on, and driving the pin high turns it off.

The diagram below Figure 28: shows the connections between the LEDs, the Arria® 10 FPGA, the HPS and the MAX® 10:

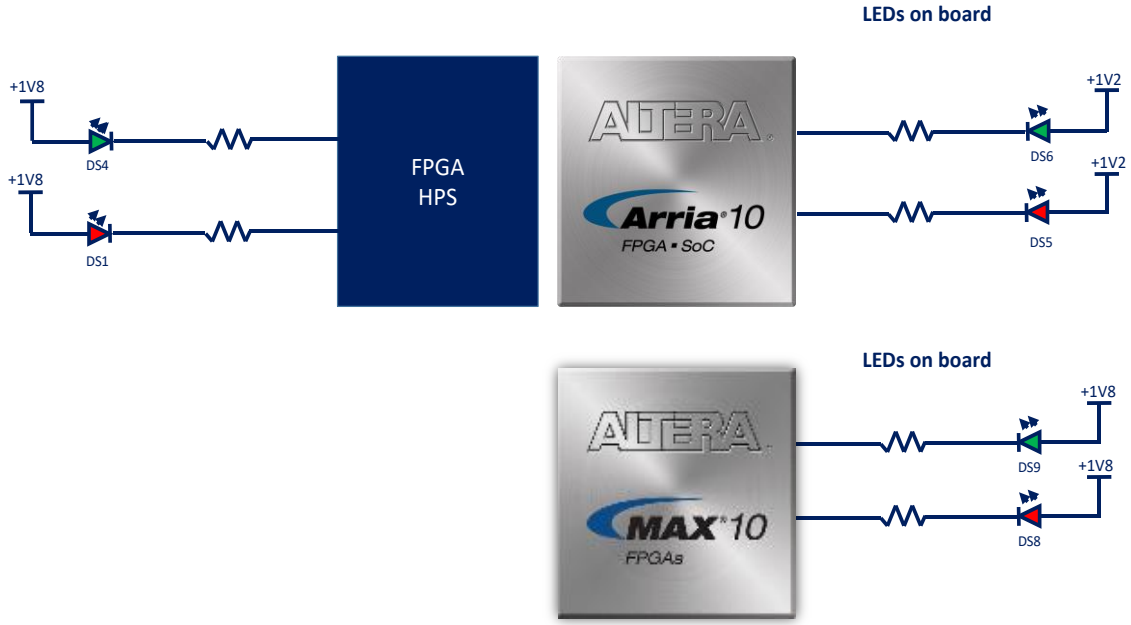



*Figure 28: LEDs Design Schematic*

| Board Reference | Schematics signal name | Description | IO standard | Connected to |
|---|---|---|---|---|
| DS4 | hps_usr_ledg | Driving a logic 1 on the I/O port, turns the LED OFF<br><br>Driving a logic 0 on the I/O port, turns the LED ON. | LVCMOS 1V8 | Arria®10 pin F15 |
| DS1 | hps_usr_ledr | | LVCMOS 1V8 | Arria®10 pin H15 |
| DS6 | fpga_usr_ledg | | LVCMOS 1V2 | Arria®10 pin J25 |
| DS5 | fpga_usr_ledr | | LVCMOS 1V2 | Arria®10 pin J26 |
| DS9 | max10_usr_ledg | | LVCMOS 1V8 | MAX®10 pin N11 |
| DS8 | max10_usr_ledr | | LVCMOS 1V8 | MAX®10 pin N12 |

*Table 6: Running User LEDs*

## 7.6 User Dipswitch

A two-position user dipswitch is connected to the MAX® 10 device.

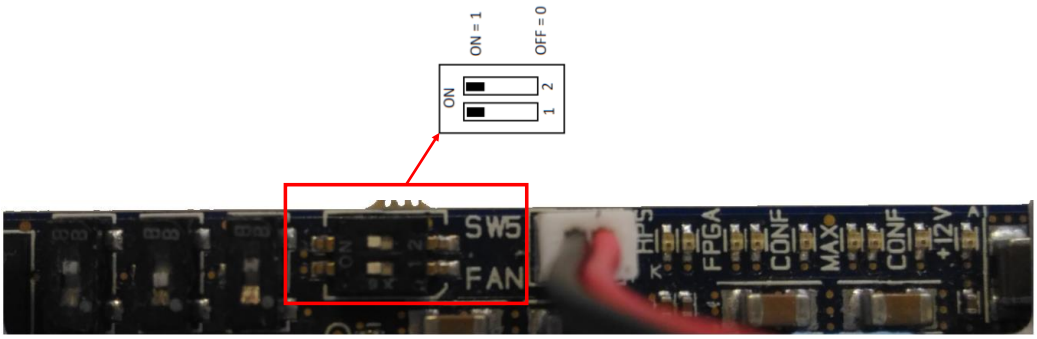



*Figure 29: User dipswitch SW5 location*

| Board Reference | Schematics signal name | Description | IO standard | MAX® 10 Pin Number |
|---|---|---|---|---|
| SW5-1 | max10_sw1 | Slide Off position = Driving a logic 0 on the I/O port | LVCMOS 1V8 | M8 |
| SW5-2 | max10_sw2 | Slide ON position = Driving a logic 1 on the I/O port | LVCMOS 1V8 | M9 |

*Table 7: User Dipswitch Configuration*

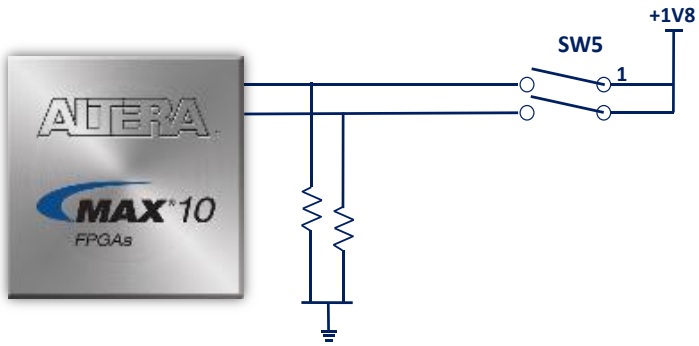



*Figure 30: User Dipswitch Design Schematic*

## 7.7 MAX® 10 FPGA connections

The detailed architecture of the MAX® 10 is described in Chapter 7.
The following table shows the pinout for MAX® 10 FPGA connector:

| Signal Name | MAX 10 Pin | Description | Pin label | IO Standard |
|---|---|---|---|---|
| clk_25mhz_cpld | G5 | CPLD Clock | IO/CLK0_n/DIFFIO_RX_L20_n(2) | LVCMOS 1V8 |
| clk_max10_n | H5 | MAX® 10 Clock | IO/CLK1_n/DIFFIO_RX_L22_n(2) | LVDS |
| clk_max10_p | H4 | | IO/CLK1_p/DIFFIO_RX_L22_p(2) | |
| com_f2m | N4 | FPGA Communication | IO/DIFFIO_RX_B4_n | LVCMOS 1V8 |
| com_m2f | K1 | FPGA Communication | IO/DIFFIO_RX_L32_n | LVCMOS 1V8 |
| emmc_rst_n | M1 | eMMC reset | IO/DIFFIO_RX_L23_n | LVCMOS 1V8 |
| fpga_rst | J1 | FPGA reset | IO/DIFFIO_RX_L21_n | LVCMOS 1V8 |
| fx3_32khz | N7 | FX3 Clock | IO/DIFFIO_RX_B6_n | LVCMOS 1V8 |
| fx3_reset_n | N8 | FX3 reset | IO/DIFFIO_RX_B6_p | LVCMOS 1V8 |
| gbe1_reset_n | M3 | Ethernet reset | IO/PLL_L_CLKOUT_n/DIFFIO_RX_L31_n(2) | LVCMOS 1V8 |
| gbe2_reset_n | G4 | Ethernet reset | IO/DIFFIO_RX_L14_p | LVCMOS 1V8 |
| hpc_pg_c2m | L12 | HPC Power | IO/DIFFIO_RX_R2_n | LVCMOS 3V3 |
| hpc_pg_m2c | L13 | HPC Power | IO | LVCMOS 3V3 |
| hpc_prsnt_m2c_n | K13 | M2C reset | IO/VREFB5N0 | LVCMOS 3V3 |
| hps_good | J2 | HPS Power | IO/DIFFIO_RX_L21_p | LVCMOS 1V8 |
| hps_por_n | K6 | HPS Power | IO/DIFFIO_TX_RX_B13_p | LVCMOS 1V8 |
| hps_rst | J1 | HPS Reset | IO/DIFFIO_RX_L21_n | LVCMOS 1V8 |
| hps_rst_n | H3 | HPS Reset | IO/DIFFIO_RX_L16_p | LVCMOS 1V8 |
| lpc_pg_c2m | H13 | LPC Power | IO/DIFFIO_RX_R13_n | LVCMOS 3V3 |
| lpc_prsnt_n_m2c | K12 | LPC Reset | IO/DIFFIO_RX_R11_p | LVCMOS 3V3 |
| max_fanpwm | B7 | PWM fan | IO/VREFB8N0 | LVCMOS 3V3 |
| max10_crc_error | D6 | MAX 10 Config. Status | IO/CRC_ERROR/DIFFIO_RX_T34_n(8) | LVCMOS 3V3 |
| max10_dev_clr_n | B9 | MAX 10 Config. Status | IO/DEV_CLRn/DIFFIO_RX_T28_n(8) | LVCMOS 3V3 |
| max10_dev_oe | D8 | MAX 10 Config. Status | IO/DEV_OE/DIFFIO_RX_T30_p(8) | LVCMOS 3V3 |
| max10_jtag_en | E5 | JTAG Enable | IO/JTAGEN(1B) | LVCMOS 1V8 |
| max10_uart_cts_n | M12 | UART Interface | IO/DIFFIO_TX_RX_B16_p | LVCMOS 1V8 |
| max10_uart_rts_n | M13 | UART Interface | IO/DIFFIO_TX_RX_B16_n | LVCMOS 1V8 |
| max10_uart_rx | M10 | UART Interface | IO/DIFFIO_TX_RX_B22_n | LVCMOS 1V8 |
| max10_uart_tx | L10 | UART Interface | IO/DIFFIO_TX_RX_B22_p | LVCMOS 1V8 |
| max10_usr_ledg | N11 | LED Interface | IO/VREFB3N0 | LVCMOS 1V8 |
| max10_usr_ledr | N12 | LED Interface | IO | LVCMOS 1V8 |
| pmbus_alert_n | B13 | PMBus Interface | IO/DIFFIO_RX_R34_p | LVCMOS 3V3 |
| pmbus_scl | C13 | PMBus Interface | IO | LVCMOS 3V3 |
| pmbus_sda | D13 | PMBus Interface | IO/VREFB6N0 | LVCMOS 3V3 |
| psa_reset_n | A6 | Power Supply Reset | IO/DIFFIO_RX_T31_n | LVCMOS 3V3 |
| psb_reset_n | A5 | Power Supply Reset | IO | LVCMOS 3V3 |

| Signal Name | MAX 10 Pin | Description | Pin label | IO Standard |
|---|---|---|---|---|
| si5341_fdec | L12 | I²C Interface SI5341 | IO | LVCMOS 1V8 |
| si5341_finc | N10 | I²C Interface SI5341 | IO/DIFFIO_RX_B17_n | LVCMOS 1V8 |
| si5341_intr_n | L4 | I²C Interface SI5341 | IO/DIFFIO_TX_RX_B1_p | LVCMOS 1V8 |
| si5341_lol_n | N9 | I²C Interface SI5341 | IO/DIFFIO_RX_B17_p | LVCMOS 1V8 |
| si5341_rst_n | J7 | I²C Interface SI5341 | IO/DIFFIO_TX_RX_B15_n | LVCMOS 1V8 |
| si5341_scl | H2 | I²C Interface SI5341 | IO/DIFFIO_RX_L16_n | LVCMOS 1V8 |
| si5341_sda | H1 | I²C Interface SI5341 | IO/VREFB1N0 | LVCMOS 1V8 |
| si5341_sync_n | K7 | I²C Interface SI5341 | IO/DIFFIO_TX_RX_B15_p | LVCMOS 1V8 |
| som_cg_c2m | G12 | Module configuration good output | IO/DIFFIO_RX_R15_n | LVCMOS 3V3 |
| som_cg_m2c | J13 | Module configuration good input | IO/DIFFIO_RX_R13_p | LVCMOS 3V3 |
| som_pg | G13 | Power supply Good | IO/DIFFIO_RX_R15_p | LVCMOS 3V3 |
| som_rst_c2m | E12 | Module reset output | IO/DIFFIO_RX_R22_n | LVCMOS 3V3 |
| som_rst_m2c | F12 | Module reset input | IO/DIFFIO_RX_R22_p | LVCMOS 3V3 |
| temp_alert | M2 | Temperature Sensor | IO/DIFFIO_RX_L23_p | LVCMOS 1V8 |
| temp_fault | A12 | Temperature Sensor | IO/DIFFIO_RX_R34_n | LVCMOS 3V3 |
| usbotg_reset_n | F4 | USB OTG Reset | IO/DIFFIO_RX_L14_n | LVCMOS 1V8 |
| CONF_DONE | C5 | Arria® 10 Config. Status | IO/CONF_DONE/DIFFIO_RX_T36_n(8) | LVCMOS 3V3 |
| CONFIG_SEL | D7 | Arria® 10 Config. Status | IO/CONFIG_SEL(8) | LVCMOS 3V3 |
| NCONFIG | E7 | Arria® 10 Config. Status | nCONFIG(8) | LVCMOS 3V3 |
| NSTATUS | C4 | Arria® 10 Config. Status | IO/nSTATUS/DIFFIO_RX_T36_p(8) | LVCMOS 3V3 |
| TDI | F5 | JTAG | IO/TDI/DIFFIO_RX_L12_n(1B) | LVCMOS 1V8 |
| TCK | G2 | JTAG | IO/TCK/DIFFIO_RX_L11_p(1B) | LVCMOS 1V8 |
| TDO | F6 | JTAG | IO/TDO/DIFFIO_RX_L12_p(1B) | LVCMOS 1V8 |
| TMS | G1 | JTAG | IO/TMS/DIFFIO_RX_L11_n(1B) | LVCMOS 1V8 |
| max10_sw1 | M8 | Dipswitch | IO/DIFFIO_RX_B14_n | LVCMOS 1V8 |
| max10_sw2 | M9 | Dipswitch | IO/DIFFIO_RX_B14_p | LVCMOS 1V8 |

*Table 8: MAX® 10 FPGA Connections*

## 7.8 PCIe-Specific Interfaces

The quick links available on the SoC SoM A10 FMC connectors offer the possibility to create two PCI interfaces; a PCIe x8 Gen3 interface without CvP and a PCIe x8 Gen3 interface with CvP.

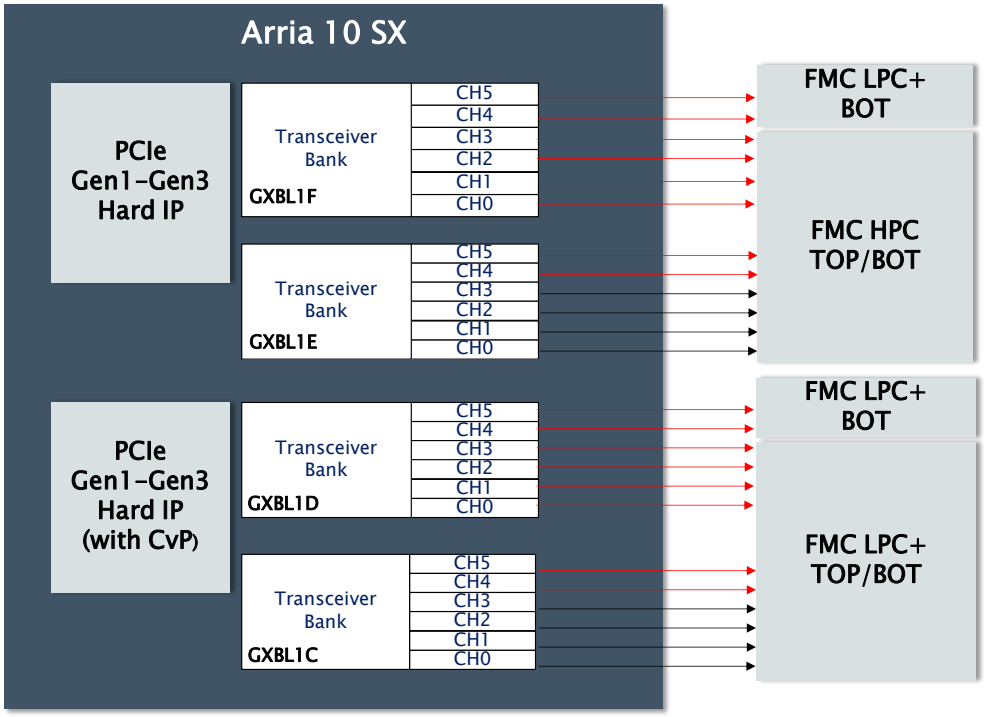



*Figure 31: PCIe-Specific Interfaces*

The table below give the correspondence between PCIe lane and signal name. Please refer to Superpinout file to know the pins connection of the Arria10 and the LPC/HPC connectors.

| PCIe Lane | A10 Bank | A10 Channel | Signal name |
|---|---|---|---|
| 0 | 1C | CH4_[RX/TX] | LPC_DP4_[M2C/C2M] |
| 1 | | CH5_[RX/TX] | LPC_DP5_[M2C/C2M] |
| 2 | 1D | CH0_[RX/TX] | LPC_DP6_[M2C/C2M] |
| 3 | | CH1_[RX/TX] | LPC_DP7_[M2C/C2M] |
| 4 | | CH2_[RX/TX] | LPC_DP8_[M2C/C2M] |
| 5 | | CH3_[RX/TX] | LPC_DP9_[M2C/C2M] |
| 6 | | CH4_[RX/TX] | LPC_DP10_[M2C/C2M] |
| 7 | | CH5_[RX/TX] | LPC_DP11_[M2C/C2M] |
| **PCIe** | **A10 Bank** | **A10 Channel** | **Signal name** |
| 0 | 1F | CH4_[RX/TX] | LPC_DP12_[M2C/C2M] |
| 1 | | CH5_[RX/TX] | LPC_DP13_[M2C/C2M] |
| 2 | 1E | CH0_[RX/TX] | HPC_DP9_[M2C/C2M] |
| 3 | | CH1_[RX/TX] | HPC_DP8_[M2C/C2M] |
| 4 | | CH2_[RX/TX] | HPC_DP7_[M2C/C2M] |
| 5 | | CH3_[RX/TX] | HPC_DP6_[M2C/C2M] |
| 6 | | CH4_[RX/TX] | HPC_DP5_[M2C/C2M] |
| 7 | | CH5_[RX/TX] | HPC_DP4_[M2C/C2M] |

These interfaces can be realized on a carrier card with a PCIe edge connector or Root port connector providing physical support for PCIe communication.

- The PCIe **endpoint** solution can be created with a hard IP core
- The PCIe **root complex** solution can be created with a soft IP core

The Hard IP for PCI Express architecture includes the option to configure the FPGA and initialize the PCI Express link, as illustrated in the diagram below.

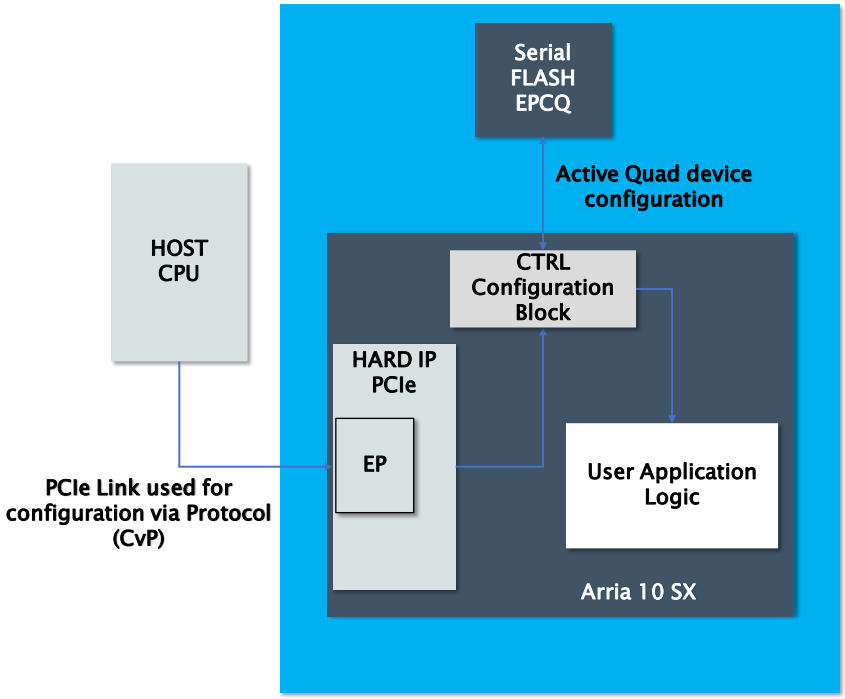



*Figure 32: PCIe Boot Mode with CvP*

A single Program Object File (.pof) programs the I/O ring and PCIe Hard IP before PCIe link training and enumeration begins.
The .pof file is divided into two parts:

- Part #1: The I/O bitstream contains the data to program the I/O ring, the Hard IP for PCI Express, and other elements that are considered part of the periphery image.
- Part #2: The core bitstream contains the data that the host CPU uses to program the FPGA fabric.

The table below shows the sizes of the Uncompressed Configuration bitstream and the IOCSR bitstream:

| Variant | Product Line | Uncompressed Configuration Bit Stream Size (bits) | IOCSR Bit Stream Size (bits) |
|---|---|---|---|
| Arria® 10 SX | SX 160 | 81,923,582 | 2,507,264 |
| | SX 220 | 81,923,582 | 2,507,264 |
| | SX 270 | 122,591,622 | 2,507,264 |
| | SX 320 | 122,591,622 | 2,507,264 |
| | SX 480 | 177,341,246 | 2,695,680 |
| | SX 570 | 252,831,072 | 2,884,096 |
| | SX 660 | 252,831,072 | 2,884,096 |

## 7.9 DDR4 FPGA interface

A 32-bit DDR4 interface is connected to the FPGA. The following table shows the pin assignments for this interface:

| Name | Pin | Pin Label | IO Standard |
|---|---|---|---|
| ddr4_a[0] | AH3 | IO12/LVDS3B_18_p/DQ58 | SSTL-12 |
| ddr4_a[1] | AG3 | IO13/LVDS3B_18_n/DQ58 | SSTL-12 |
| ddr4_a[2] | AG6 | IO14/LVDS3B_17_p/DQ58 | SSTL-12 |
| ddr4_a[3] | AF6 | IO15/LVDS3B_17_n/DQ58 | SSTL-12 |
| ddr4_a[4] | AF1 | IO16/LVDS3B_16_p/DQS58/CQ58 | SSTL-12 |
| ddr4_a[5] | AE1 | IO17/LVDS3B_16_n/DQSn58/CQn58 | SSTL-12 |
| ddr4_a[6] | AF4 | IO18/LVDS3B_15_p/PLL_CLKOUT0_p/FB0/DQ58(3B) | SSTL-12 |
| ddr4_a[7] | AF3 | IO19/LVDS3B_15_n/PLL_CLKOUT0_n/DQ58(3B) | SSTL-12 |
| ddr4_a[8] | AG5 | IO20/LVDS3B_14_p/DQ58 | SSTL-12 |
| ddr4_a[9] | AF5 | IO21/LVDS3B_14_n/DQ58 | SSTL-12 |
| ddr4_a[10] | AE2 | IO22/LVDS3B_13_p/CLK0_p/DQ58(3B) | SSTL-12 |
| ddr4_a[11] | AE3 | IO23/LVDS3B_13_n/CLK0_n/DQ58(3B) | SSTL-12 |
| ddr4_a[12] | AE7 | IO27/LVDS3B_11_n/DQ57 | SSTL-12 |
| ddr4_a[13] | AC8 | IO28/LVDS3B_10_p/PLL_CLKOUT1_p/FB1/DQS57/CQ57(3B) | SSTL-12 |
| ddr4_a[14] | AD9 | IO29/LVDS3B_10_n/PLL_CLKOUT1_n/DQSn57/CQn57(3B) | SSTL-12 |
| ddr4_a[15] | AC9 | IO30/LVDS3B_9_p/DQ57 | SSTL-12 |
| ddr4_a[16] | AC10 | IO31/LVDS3B_9_n/DQ57 | SSTL-12 |
| ddr4_act_n[0] | AJ2 | IO3/LVDS3B_23_n/DQ59 | SSTL-12 |
| ddr4_alert_n | AH4 | IO7/LVDS3B_21_n/DQ59 | SSTL-12 |
| ddr4_ba[0] | AD6 | IO33/LVDS3B_8_n/DQ57 | SSTL-12 |
| ddr4_ba[1] | AD7 | IO34/LVDS3B_7_p/DQ57 | SSTL-12 |
| ddr4_bg[0] | AC7 | IO35/LVDS3B_7_n/DQ57 | SSTL-12 |
| ddr4_bg[1] | AL1 | IO0/LVDS3B_24_p/DQ59 | SSTL-12 |
| ddr4_cke[0] | AJ4 | IO6/LVDS3B_21_p/DQ59 | SSTL-12 |
| ddr4_ck_n[0] | AH2 | IO9/LVDS3B_20_n/DQ59 | Differential 1.2-V SSTL |
| ddr4_ck[0] | AJ1 | IO8/LVDS3B_20_p/DQ59 | Differential 1.2-V SSTL |
| ddr4_cs_n[0] | AK2 | IO2/LVDS3B_23_p/DQ59 | SSTL-12 |
| ddr4_dbi_n[0] | AM6 | IO14/LVDS3A_17_p/DQ62 | 1.2-V POD |
| ddr4_dbi_n[1] | AN5 | IO11/LVDS3A_19_n/DQ63 | 1.2-V POD |
| ddr4_dbi_n[2] | AE9 | IO47/LVDS3A_1_n/DQ60 | 1.2-V POD |
| ddr4_dbi_n[3] | AD1 | IO45/LVDS3B_2_n/DQ56 | 1.2-V POD |
| ddr4_dq[0] | AM3 | IO18/LVDS3A_15_p/PLL_CLKOUT0_p/FB0/DQ62(3A) | 1.2-V POD |
| ddr4_dq[1] | AL6 | IO20/LVDS3A_14_p/DQ62 | 1.2-V POD |
| ddr4_dq[10] | AP6 | IO9/LVDS3A_20_n/DQ63 | 1.2-V POD |
| ddr4_dq[11] | AL9 | IO2/LVDS3A_23_p/DQ63 | 1.2-V POD |
| ddr4_dq[12] | AP5 | IO10/LVDS3A_19_p/DQ63 | 1.2-V POD |
| ddr4_dq[13] | AP7 | IO8/LVDS3A_20_p/DQ63 | 1.2-V POD |

| Name | Pin | Pin Label | IO Standard |
|---|---|---|---|
| ddr4_dq[14] | AM8 | IO7/LVDS3A_21_n/DQ63 | 1.2-V POD |
| ddr4_dq[15] | AM7 | IO1/LVDS3A_24_n/DQ63 | 1.2-V POD |
| ddr4_dq[16] | AH10 | IO42/LVDS3A_3_p/DQ60 | 1.2-V POD |
| ddr4_dq[17] | AE8 | IO45/LVDS3A_2_n/DQ60 | 1.2-V POD |
| ddr4_dq[18] | AG11 | IO39/LVDS3A_5_n/DQ60 | 1.2-V POD |
| ddr4_dq[19] | AF9 | IO46/LVDS3A_1_p/DQ60 | 1.2-V POD |
| ddr4_dq[2] | AL3 | IO19/LVDS3A_15_n/PLL_CLKOUT0_n/DQ62(3A) | 1.2-V POD |
| ddr4_dq[20] | AH9 | IO43/LVDS3A_3_n/DQ60 | 1.2-V POD |
| ddr4_dq[21] | AE11 | IO37/LVDS3A_6_n/DQ60 | 1.2-V POD |
| ddr4_dq[22] | AF8 | IO44/LVDS3A_2_p/DQ60 | 1.2-V POD |
| ddr4_dq[23] | AF11 | IO38/LVDS3A_5_p/DQ60 | 1.2-V POD |
| ddr4_dq[24] | AB5 | IO39/LVDS3B_5_n/DQ56 | 1.2-V POD |
| ddr4_dq[25] | AB11 | IO47/LVDS3B_1_n/DQ56 | 1.2-V POD |
| ddr4_dq[26] | AD4 | IO43/LVDS3B_3_n/DQ56 | 1.2-V POD |
| ddr4_dq[27] | AB6 | IO38/LVDS3B_5_p/DQ56 | 1.2-V POD |
| ddr4_dq[28] | AD2 | IO44/LVDS3B_2_p/DQ56 | 1.2-V POD |
| ddr4_dq[29] | AB10 | IO46/LVDS3B_1_p/DQ56 | 1.2-V POD |
| ddr4_dq[3] | AM5 | IO15/LVDS3A_17_n/DQ62 | 1.2-V POD |
| ddr4_dq[30] | AE4 | IO42/LVDS3B_3_p/DQ56 | 1.2-V POD |
| ddr4_dq[31] | AC5 | IO37/LVDS3B_6_n/DQ56 | 1.2-V POD |
| ddr4_dq[4] | AP4 | IO12/LVDS3A_18_p/DQ62 | 1.2-V POD |
| ddr4_dq[5] | AK6 | IO21/LVDS3A_14_n/DQ62 | 1.2-V POD |
| ddr4_dq[6] | AN4 | IO13/LVDS3A_18_n/DQ62 | 1.2-V POD |
| ddr4_dq[7] | AL4 | IO23/LVDS3A_13_n/CLK0_n/DQ62(3A) | 1.2-V POD |
| ddr4_dq[8] | AL8 | IO3/LVDS3A_23_n/DQ63 | 1.2-V POD |
| ddr4_dq[9] | AN8 | IO6/LVDS3A_21_p/DQ63 | 1.2-V POD |
| ddr4_dqs[0](n) | AM2 | IO17/LVDS3A_16_n/DQSn62/CQn62 | Differential 1.2-V POD |
| ddr4_dqs[0] | AM1 | IO16/LVDS3A_16_p/DQS62/CQ62 | Differential 1.2-V POD |
| ddr4_dqs[1](n) | AN9 | IO5/LVDS3A_22_n/DQSn63/CQn63 | Differential 1.2-V POD |
| ddr4_dqs[1] | AP9 | IO4/LVDS3A_22_p/DQS63/CQ63 | Differential 1.2-V POD |
| ddr4_dqs[2](n) | AF10 | IO41/LVDS3A_4_n/DQSn60/CQn60 | Differential 1.2-V POD |
| ddr4_dqs[2] | AG10 | IO40/LVDS3A_4_p/DQS60/CQ60 | Differential 1.2-V POD |
| ddr4_dqs[3](n) | AB7 | IO41/LVDS3B_4_n/DQSn56/CQn56 | Differential 1.2-V POD |
| ddr4_dqs[3] | AB8 | IO40/LVDS3B_4_p/DQS56/CQ56 | Differential 1.2-V POD |
| ddr4_odt[0] | AK4 | IO4/LVDS3B_22_p/DQS59/CQ59 | SSTL-12 |
| ddr4_par[0] | AG2 | IO11/LVDS3B_19_n/DQ59 | SSTL-12 |
| ddr4_reset_n[0] | AK1 | IO1/LVDS3B_24_n/DQ59 | 1.2-V |

*Table 9: DDR4 FPGA Connections*

## 7.10 DDR4 HPS Interface

A 32-bit DDR4 interface is also connected to the HPS. The following table shows the pin assignments for this interface.

| Signal | Pin | Pin Label | IO Standard |
|---|---|---|---|
| hps_ddr4_a[0] | H27 | IO12/LVDS2K_18_p/DQ6 | SSTL-12 |
| hps_ddr4_a[1] | G27 | IO13/LVDS2K_18_n/DQ6 | SSTL-12 |
| hps_ddr4_a[2] | G23 | IO14/LVDS2K_17_p/DQ6 | SSTL-12 |
| hps_ddr4_a[3] | G22 | IO15/LVDS2K_17_n/DQ6 | SSTL-12 |
| hps_ddr4_a[4] | H25 | IO16/LVDS2K_16_p/DQS6/CQ6 | SSTL-12 |
| hps_ddr4_a[5] | G25 | IO17/LVDS2K_16_n/DQSn6/CQn6 | SSTL-12 |
| hps_ddr4_a[6] | H24 | IO18/LVDS2K_15_p/PLL_CLKOUT0_p/FB0/DQ6(2K) | SSTL-12 |
| hps_ddr4_a[7] | H23 | IO19/LVDS2K_15_n/PLL_CLKOUT0_n/DQ6(2K) | SSTL-12 |
| hps_ddr4_a[8] | H22 | IO20/LVDS2K_14_p/DQ6 | SSTL-12 |
| hps_ddr4_a[9] | J22 | IO21/LVDS2K_14_n/DQ6 | SSTL-12 |
| hps_ddr4_a[10] | G26 | IO22/LVDS2K_13_p/CLK0_p/DQ6(2K) | SSTL-12 |
| hps_ddr4_a[11] | F26 | IO23/LVDS2K_13_n/CLK0_n/DQ6(2K) | SSTL-12 |
| hps_ddr4_a[12] | F24 | IO27/LVDS2K_11_n/DQ5 | SSTL-12 |
| hps_ddr4_a[13] | E27 | IO28/LVDS2K_10_p/PLL_CLKOUT1_p/FB1/DQS5/CQ5(2K) | SSTL-12 |
| hps_ddr4_a[14] | D27 | IO29/LVDS2K_10_n/PLL_CLKOUT1_n/DQSn5/CQn5(2K) | SSTL-12 |
| hps_ddr4_a[15] | E22 | IO30/LVDS2K_9_p/DQ5 | SSTL-12 |
| hps_ddr4_a[16] | F23 | IO31/LVDS2K_9_n/DQ5 | SSTL-12 |
| hps_ddr4_act_n[0] | J24 | IO3/LVDS2K_23_n/DQ7 | SSTL-12 |
| hps_ddr4_alert_n[0] | AN20 | IO12/LVDS2J_18_p/DQ10 | SSTL-12 |
| hps_ddr4_ba[0] | D26 | IO33/LVDS2K_8_n/DQ5 | SSTL-12 |
| hps_ddr4_ba[1] | D25 | IO34/LVDS2K_7_p/DQ5 | SSTL-12 |
| hps_ddr4_bg[0] | C25 | IO35/LVDS2K_7_n/DQ5 | SSTL-12 |
| hps_ddr4_bg[1] | M24 | IO0/LVDS2K_24_p/DQ7 | SSTL-12 |
| hps_ddr4_cke[0] | J27 | IO6/LVDS2K_21_p/DQ7 | SSTL-12 |
| hps_ddr4_ck_n[0] | M23 | IO9/LVDS2K_20_n/DQ7 | Differential 1.2-V SSTL |
| hps_ddr4_ck[0] | L23 | IO8/LVDS2K_20_p/DQ7 | Differential 1.2-V SSTL |
| hps_ddr4_cs_n[0] | K24 | IO2/LVDS2K_23_p/DQ7 | SSTL-12 |
| hps_ddr4_dbi_n[0] | AP24 | IO23/LVDS2J_13_n/CLK0_n/DQ10(2J) | 1.2-V POD |
| hps_ddr4_dbi_n[1] | AE23 | IO11/LVDS2J_19_n/DQ11 | 1.2-V POD |
| hps_ddr4_dbi_n[2] | AM27 | IO35/LVDS2J_7_n/DQ9 | 1.2-V POD |
| hps_ddr4_dbi_n[3] | AD25 | IO47/LVDS2J_1_n/DQ8 | 1.2-V POD |
| hps_ddr4_dq[0] | AP21 | IO18/LVDS2J_15_p/PLL_CLKOUT0_p/FB0/DQ10(2J) | 1.2-V POD |
| hps_ddr4_dq[1] | AN24 | IO22/LVDS2J_13_p/CLK0_p/DQ10(2J) | 1.2-V POD |
| hps_ddr4_dq[2] | AP22 | IO19/LVDS2J_15_n/PLL_CLKOUT0_n/DQ10(2J) | 1.2-V POD |
| hps_ddr4_dq[3] | AL24 | IO20/LVDS2J_14_p/DQ10 | 1.2-V POD |

| Signal | Pin | Pin Label | IO Standard |
|---|---|---|---|
| hps_ddr4_dq[4] | AP20 | IO13/LVDS2J_18_n/DQ10 | 1.2-V POD |
| hps_ddr4_dq[5] | AL25 | IO21/LVDS2J_14_n/DQ10 | 1.2-V POD |
| hps_ddr4_dq[6] | AM23 | IO14/LVDS2J_17_p/DQ10 | 1.2-V POD |
| hps_ddr4_dq[7] | AN23 | IO15/LVDS2J_17_n/DQ10 | 1.2-V POD |
| hps_ddr4_dq[8] | AK23 | IO3/LVDS2J_23_n/DQ11 | 1.2-V POD |
| hps_ddr4_dq[9] | AF24 | IO10/LVDS2J_19_p/DQ11 | 1.2-V POD |
| hps_ddr4_dq[10] | AK22 | IO1/LVDS2J_24_n/DQ11 | 1.2-V POD |
| hps_ddr4_dq[11] | AK24 | IO7/LVDS2J_21_n/DQ11 | 1.2-V POD |
| hps_ddr4_dq[12] | AF23 | IO8/LVDS2J_20_p/DQ11 | 1.2-V POD |
| hps_ddr4_dq[13] | AJ24 | IO6/LVDS2J_21_p/DQ11 | 1.2-V POD |
| hps_ddr4_dq[14] | AG23 | IO9/LVDS2J_20_n/DQ11 | 1.2-V POD |
| hps_ddr4_dq[15] | AL23 | IO2/LVDS2J_23_p/DQ11 | 1.2-V POD |
| hps_ddr4_dq[16] | AP26 | IO32/LVDS2J_8_p/DQ9 | 1.2-V POD |
| hps_ddr4_dq[17] | AL26 | IO25/LVDS2J_12_n/CLK1_n/DQ9(2J) | 1.2-V POD |
| hps_ddr4_dq[18] | AP27 | IO33/LVDS2J_8_n/DQ9 | 1.2-V POD |
| hps_ddr4_dq[19] | AK26 | IO31/LVDS2J_9_n/DQ9 | 1.2-V POD |
| hps_ddr4_dq[20] | AP25 | IO26/LVDS2J_11_p/RZQ_2J/DQ9 | 1.2-V POD |
| hps_ddr4_dq[21] | AK27 | IO30/LVDS2J_9_p/DQ9 | 1.2-V POD |
| hps_ddr4_dq[22] | AN25 | IO27/LVDS2J_11_n/DQ9 | 1.2-V POD |
| hps_ddr4_dq[23] | AN27 | IO34/LVDS2J_7_p/DQ9 | 1.2-V POD |
| hps_ddr4_dq[24] | AJ27 | IO44/LVDS2J_2_p/DQ8 | 1.2-V POD |
| hps_ddr4_dq[25] | AD24 | IO37/LVDS2J_6_n/DQ8 | 1.2-V POD |
| hps_ddr4_dq[26] | AJ26 | IO42/LVDS2J_3_p/DQ8 | 1.2-V POD |
| hps_ddr4_dq[27] | AH27 | IO45/LVDS2J_2_n/DQ8 | 1.2-V POD |
| hps_ddr4_dq[28] | AJ25 | IO38/LVDS2J_5_p/DQ8 | 1.2-V POD |
| hps_ddr4_dq[29] | AE24 | IO46/LVDS2J_1_p/DQ8 | 1.2-V POD |
| hps_ddr4_dq[30] | AH25 | IO39/LVDS2J_5_n/DQ8 | 1.2-V POD |
| hps_ddr4_dq[31] | AH26 | IO43/LVDS2J_3_n/DQ8 | 1.2-V POD |
| hps_ddr4_dqs_n[0] | AN22 | IO17/LVDS2J_16_n/DQSn10/CQn10 | Differential 1.2-V POD |
| hps_ddr4_dqs[0] | AM22 | IO16/LVDS2J_16_p/DQS10/CQ10 | Differential 1.2-V POD |
| hps_ddr4_dqs_n[1] | AH24 | IO5/LVDS2J_22_n/DQSn11/CQn11 | Differential 1.2-V POD |
| hps_ddr4_dqs[1] | AH23 | IO4/LVDS2J_22_p/DQS11/CQ11 | Differential 1.2-V POD |
| hps_ddr4_dqs_n[2] | AM26 | IO29/LVDS2J_10_n/PLL_CLKOUT1_n/DQSn9/CQn9(2J) | Differential 1.2-V POD |
| hps_ddr4_dqs[2] | AM25 | IO28/LVDS2J_10_p/PLL_CLKOUT1_p/FB1/DQS9/CQ9(2J) | Differential 1.2-V POD |
| hps_ddr4_dqs_n[3] | AF25 | IO41/LVDS2J_4_n/DQSn8/CQn8 | Differential 1.2-V POD |

| Signal | Pin | Pin Label | IO Standard |
|---|---|---|---|
| hps_ddr4_dqs[3] | AG25 | IO40/LVDS2J_4_p/DQS8/CQ8 | Differential 1.2-V POD |
| hps_ddr4_odt[0] | K25 | IO4/LVDS2K_22_p/DQS7/CQ7 | SSTL-12 |
| hps_ddr4_par[0] | K22 | IO11/LVDS2K_19_n/DQ7 | SSTL-12 |
| hps_ddr4_reset_n[0] | L24 | IO1/LVDS2K_24_n/DQ7 | 1.2 V |

*Table 10: DDR4 HPS Connections*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 7.11 FX3 interface

The following table shows pin assignments for the FX3 interface:

| Signal | Pin | Pin Label | IO Standard |
|---|---|---|---|
| fx3_a[0] | AK21 | IO30/LVDS2I_9_p/DQ13 | 1.2 V |
| fx3_a[1] | AM21 | IO29/LVDS2I_10_n/PLL_CLKOUT1_n/DQSn13/CQn13(2I) | 1.2 V |
| fx3_pclk | AC4 | IO36/LVDS3B_6_p/DQ56 | 1.2 V |
| fx3_dq[0] | AL19 | IO33/LVDS2I_8_n/DQ13 | 1.2 V |
| fx3_dq[1] | AN19 | IO34/LVDS2I_7_p/DQ13 | 1.2 V |
| fx3_dq[2] | AP19 | IO35/LVDS2I_7_n/DQ13 | 1.2 V |
| fx3_dq[3] | AM20 | IO27/LVDS2I_11_n/DQ13 | 1.2 V |
| fx3_dq[4] | AJ19 | IO25/LVDS2I_12_n/CLK1_n/DQ13(2I) | 1.2 V |
| fx3_dq[5] | AK19 | IO32/LVDS2I_8_p/DQ13 | 1.2 V |
| fx3_dq[6] | AL20 | IO26/LVDS2I_11_p/RZQ_2I/DQ13 | 1.2 V |
| fx3_dq[7] | AL21 | IO28/LVDS2I_10_p/PLL_CLKOUT1_p/FB1/DQS13/CQ13(2I) | 1.2 V |
| fx3_flaga | B26 | IO44/LVDS2K_2_p/DQ4 | 1.2 V |
| fx3_flagb | C24 | IO40/LVDS2K_4_p/DQS4/CQ4 | 1.2 V |
| fx3_flagc | B23 | IO46/LVDS2K_1_p/DQ4 | 1.2 V |
| fx3_flagd | C23 | IO47/LVDS2K_1_n/DQ4 | 1.2 V |
| fx3_pktend_n | AJ6 | IO27/LVDS3A_11_n/DQ61 | 1.2 V |
| fx3_slcs_n | AD5 | IO32/LVDS3B_8_p/DQ57 | 1.2 V |
| fx3_sloe_n | AJ21 | IO31/LVDS2I_9_n/DQ13 | 1.2 V |
| fx3_slrd_n | AJ20 | IO24/LVDS2I_12_p/CLK1_p/DQ13(2I) | 1.2 V |
| fx3_slwr_n | AE6 | IO26/LVDS3B_11_p/RZQ_3B/DQ57 | 1.2 V |

*Table 11: FX3 Connections*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 7.12 FMC HPC TOP interface

### 7.12.1 FMC Capabilities

The full FMC-HPC interface features:

- Ten Rx/Tx transceivers from the FPGA up to 10.3125 Gb/s
- Two transceiver clock inputs and 80 LVDS pairs (usable as 160 LVCMOS18 single-ended signals)
- Four LVDS clock signals
- JTAG configuration signals
- +12V/+3.3V/VADJ (1.8V) power supplies

The Achilles DevKit FMC interfaces on the top side conform to ANSI VITA57 mechanical requirements, so any specification-compliant FMC mezzanine card can be mounted on the Achilles board Top connectors.

### 7.12.2 FMC Mechanical Capabilities

The following diagram shows the mechanical dimensions of the **air-cooled** FMC Mezzanine card compatible with the Achilles DevKit. Please refer to the VITA57_AV57DOT1 to obtain more information on the FMC/VITA57.1 interface or contact REFLEX CES to help you develop or find the best FMC module for your application.

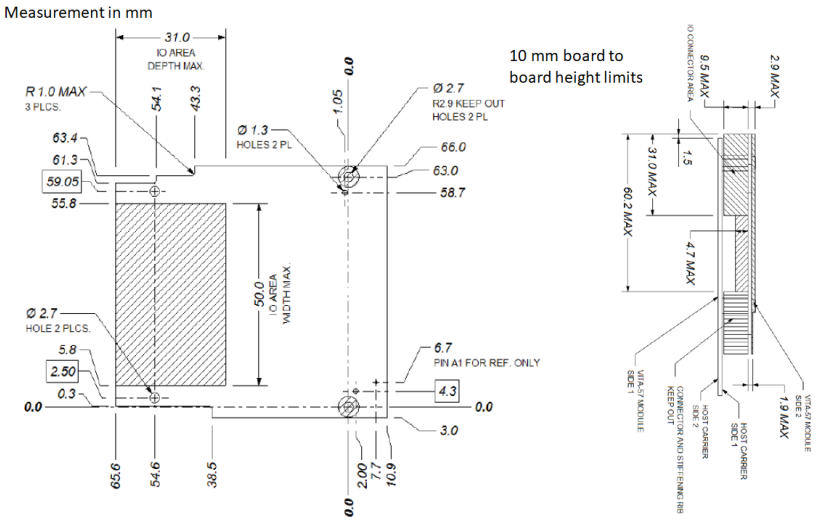



Figure 33: Mechanical Dimensions of the FMC-HPC card

Note that the Achilles DevKit is also compliant with FMC/VITA57.1 for mounting FMC conduction-cooled cards.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

### 7.12.3 FMC Modules Capabilities

The Achilles Top connectors are ASP-134486-01 from SAMTEC (High Speed FMC HPC 10x40pts Straight Female connector Cms **h=6.68 mm)**

The Arria® 10 FPGA **doesn't allow** for full VITA57.1 compliance on the LVDS and LVCMOS FMC interfaces as its IOs are limited to 1.8V.

The table below shows the power supply from the module to the FMC interfaces:

| Voltage Supply | Carrier board Voltage Range | Number of Pins | Total Max Amps |
|---|---|---|---|
| VADJ | +1.8V | 4 | 4A |
| 3P3VAUX | +3.3V | 1 | 20mA |
| 3P3V | +3.3V | 4 | 3A |
| 12P0V | +12V | 2 | 1A |

*Table 12: FMC Power Capabilities*

**HPC connector compliant VITA57.1 / +1V8 limitation**

| | K | J | H | G | F | E | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | VREF_B_M2C | GND | VREF_A_M2C | GND | PG_M2C | GND | PG_C2M | GND | CLK_DIR | GND |
| 2 | GND | CLK3_BIDIR_P | PRSNT_M2C_L | CLK1_M2C_P | GND | HA01_P_CC | GND | DP0_C2M_P | GND | DP1_M2C_P |
| 3 | GND | CLK3_BIDIR_N | GND | CLK1_M2C_N | GND | HA01_N_CC | GND | DP0_C2M_N | GND | DP1_M2C_N |
| 4 | CLK2_BIDIR_P | GND | CLK0_M2C_P | GND | HA00_P_CC | GND | GBTCLK0_M2C_P | GND | DP9_M2C_P | GND |
| 5 | CLK2_BIDIR_N | GND | CLK0_M2C_N | GND | HA00_N_CC | GND | GBTCLK0_M2C_N | GND | DP9_M2C_N | GND |
| 6 | GND | HA03_P | GND | LA00_P_CC | GND | HA05_P | GND | DP0_M2C_P | GND | DP2_M2C_P |
| 7 | HA02_P | HA03_N | LA02_P | LA00_N_CC | HA04_P | HA05_N | GND | DP0_M2C_N | GND | DP2_M2C_N |
| 8 | HA02_N | GND | LA02_N | GND | HA04_N | GND | LA01_P_CC | GND | DP8_M2C_P | GND |
| 9 | GND | HA07_P | GND | LA03_P | GND | HA09_P | LA01_N_CC | GND | DP8_M2C_N | GND |
| 10 | HA06_P | HA07_N | LA04_P | LA03_N | HA08_P | HA09_N | GND | LA06_P | GND | DP3_M2C_P |
| 11 | HA06_N | GND | LA04_N | GND | HA08_N | GND | LA05_P | LA06_N | GND | DP3_M2C_N |
| 12 | GND | HA11_P | GND | LA08_P | GND | HA13_P | LA05_N | GND | DP7_M2C_P | GND |
| 13 | HA10_P | HA11_N | LA07_P | LA08_N | HA12_P | HA13_N | GND | GND | DP7_M2C_N | GND |
| 14 | HA10_N | GND | LA07_N | GND | HA12_N | GND | LA09_P | LA10_P | GND | DP4_M2C_P |
| 15 | GND | HA14_P | GND | LA12_P | GND | HA16_P | LA09_N | LA10_N | GND | DP4_M2C_N |
| 16 | HA17_P_CC | HA14_N | LA11_P | LA12_N | HA15_P | HA16_N | GND | GND | DP6_M2C_P | GND |
| 17 | HA17_N_CC | GND | LA11_N | GND | HA15_N | GND | LA13_P | GND | DP6_M2C_N | GND |
| 18 | GND | HA18_P | GND | LA16_P | GND | HA20_P | LA13_N | LA14_P | GND | DP5_M2C_P |
| 19 | HA21_P | HA18_N | LA15_P | LA16_N | HA19_P | HA20_N | GND | LA14_N | GND | DP5_M2C_N |
| 20 | HA21_N | GND | LA15_N | GND | HA19_N | GND | LA17_P_CC | GND | GBTCLK1_M2C_P | GND |
| 21 | GND | HA22_P | GND | LA20_P | GND | HB03_P | LA17_N_CC | GND | GBTCLK1_M2C_N | GND |
| 22 | HA23_P | HA22_N | LA19_P | LA20_N | HB02_P | HB03_N | GND | LA18_P_CC | GND | DP1_C2M_P |
| 23 | HA23_N | GND | LA19_N | GND | HB02_N | GND | LA23_P | LA18_N_CC | GND | DP1_C2M_N |
| 24 | GND | HB01_P | GND | LA22_P | GND | HB05_P | LA23_N | GND | DP9_C2M_P | GND |
| 25 | HB00_P_CC | HB01_N | LA21_P | LA22_N | HB04_P | HB05_N | GND | GND | DP9_C2M_N | GND |
| 26 | HB00_N_CC | GND | LA21_N | GND | HB04_N | GND | LA26_P | LA27_P | GND | DP2_C2M_P |
| 27 | GND | HB07_P | GND | LA25_P | GND | HB09_P | LA26_N | LA27_N | GND | DP2_C2M_N |
| 28 | HB06_P_CC | HB07_N | LA24_P | LA25_N | HB08_P | HB09_N | GND | GND | DP8_C2M_P | GND |
| 29 | HB06_N_CC | GND | LA24_N | GND | HB08_N | GND | TCK | GND | DP8_C2M_N | GND |
| 30 | GND | HB11_P | GND | LA29_P | GND | HB13_P | TDI | SCL | GND | DP3_C2M_P |
| 31 | HB10_P | HB11_N | LA28_P | LA29_N | HB12_P | HB13_N | TDO | SDA | GND | DP3_C2M_N |
| 32 | HB10_N | GND | LA28_N | GND | HB12_N | GND | 3P3VAUX | GND | DP7_C2M_P | GND |
| 33 | GND | HB15_P | GND | LA31_P | GND | HB19_P | TMS | GND | DP7_C2M_N | GND |
| 34 | HB14_P | HB15_N | LA30_P | LA31_N | HB16_P | HB19_N | TRST_L | GA0 | GND | DP4_C2M_P |
| 35 | HB14_N | GND | LA30_N | GND | HB16_N | GND | GA1 | 12P0V | GND | DP4_C2M_N |
| 36 | GND | HB18_P | GND | LA33_P | GND | HB21_P | 3P3V | GND | DP6_C2M_P | GND |
| 37 | HB17_P_CC | HB18_N | LA32_P | LA33_N | HB20_P | HB21_N | GND | 12P0V | DP6_C2M_N | GND |
| 38 | HB17_N_CC | GND | LA32_N | GND | HB20_N | GND | 3P3V | GND | GND | DP5_C2M_P |
| 39 | GND | VIO_B_M2C | GND | VADJ | GND | VADJ | GND | 3P3V | GND | DP5_C2M_N |
| 40 | VIO_B_M2C | GND | VADJ | GND | VADJ | GND | 3P3V | GND | NC | GND |
| | | | LPC Connector | LPC Connector | | | LPC Connector | LPC Connector | | |

*Table 13: FMC Interface Pinout Compatibility*

## 7.13 10/100/1000 Ethernet Copper Solution

Two Ethernet 10/100/1000 Mbps PHY/RGMII Transceiver Ethernet / Micrel KSZ9031RNXCA devices are connected to the Arria® 10 on the core fabric part of the FPGA.

The following table shows pin assignments for the Ethernet PHY interface:

| Name | Pin | Pin Label | Description | IO Standard |
|---|---|---|---|---|
| gbe1_mdc | L19 | IO9/ADQ5/USB0_DATA5/SDMMC_DATA7/SPIM1_MOSI/DQ3 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_mdio | K19 | IO8/ADQ4/USB0_DATA4/SDMMC_DATA6/SPIM1_CLK/DQ3 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rx_clk | G20 | IO26/RZQ_2L/WE_N/UART0_TX/EMAC1_RX_CLK/S1_MISO/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rx_ctrl | F20 | IO27/RE_N/UART0_RX/EMAC1_RX_CTL/SPIM1_SS0_N/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rxd0 | F19 | IO30/ADQ3/UART1_TX/EMAC1_RX0/S1_SS0_N/I2C1_SDA/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rxd1 | E19 | IO31/CLE/UART1_RX/EMAC1_RXD1/SP1_MISO/I2C1_SCL/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rxd2 | C20 | IO34/ADQ6/EMAC1_RXD2/SPIS0_SS0_N/EMAC0_MDDQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_rxd3 | D20 | IO35/ADQ7/EMAC1_RXD3/SPIS0_MISO/EMAC0_MDC/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_tx_clk_a | E17 | IO24/CLK1_p/ADQ0/UART0_CTS_N/EMAC1_TX_CLK/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_tx_ctrl | E18 | IO25/CLK1_n/ADQ1/UART0_RTS_N/EMAC1_TX_CTL/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_txd0 | E21 | IO28/PLL_CLKOUT1_p/FB1/WP_N/UART1_CTS_N/DQS1/CQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_txd1 | D21 | IO29/PLL_CLK1_n/ADQ2/UART1_RTS_N/DQSn1/CQn1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_txd2 | D22 | IO32/ADQ4/EMAC1_TXD2/SPIS0_CLK/EMAC2_MDIO/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe1_txd3 | C22 | IO33/ADQ5/EMAC1_TXD3/SPIS0_MOSI/EMAC2_MDC/DQ1 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_mdc | J21 | IO7/CLE/UART1_RX/USB0_DATA3/SDMMC_DATA5/DQ3 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_mdio | K21 | IO6/ADQ3/UART1_TX/USB0_DATA2/SDMMC_DATA4/DQ3 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |

| Name | Pin | Pin Label | Description | IO Standard |
|---|---|---|---|---|
| gbe2_rx_clk | A18 | IO38/CE_N/UART1_TX/EMAC2_RX_CLK/SDMMC_CCLK/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_rx_ctrl | B18 | IO39/UART1_RX/Tra_CLK/EMAC2_RX_CTL/SDMMC_DATA1/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_rxd0 | B22 | IO42/ADQ10/EMAC2_RXD0/SDMMC_DATA4/SPIM0_MISO/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_rxd1 | A21 | IO43/ADQ11/Trace_CLK/EMAC2_RXD1/SDMMC_DATA5/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_rxd2 | C19 | IO46/ADQ14/Trace_D2/EMAC2_RXD2/SPIM0_MISO/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_rxd3 | D19 | IO47/ADQ15/Trace_D3/EMAC2_RXD3/SPIM0_SS0_N/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_tx_a_clk | C18 | IO36/ALE/EMAC2_TX_CLK/SDMMC_DATA0/I2C1_SDA/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_tx_ctrl | D17 | IO37/RB/EMAC2_TX_CTL/SDMMC_CMD/I2C1_SCL/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_txd0 | A19 | IO40/ADQ8/UART1_CTS_N/QSPI_SS2/EMAC2_TXD0/DQS0/CQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_txd1 | A20 | IO41/ADQ9/UART1_RTS_N/QSPI_SS3/EMC2_TX1/DQSn0/CQn0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_txd2 | B21 | IO44/ADQ12/Trace_D0/EMAC2_TXD2/SDMMC_DATA6/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |
| gbe2_txd3 | B20 | IO45/ADQ13/Trace_D1/EMAC2_TXD3/SDMMC_DATA7/DQ0 | Gigabit Ethernet RGMII link | LVCMOS 1V8 |

*Table 14: Ethernet Connections*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 7.14 MAX® 10 System Controller

### 7.14.1 Architecture

The following synoptic shows the MAX® 10 interfaces:

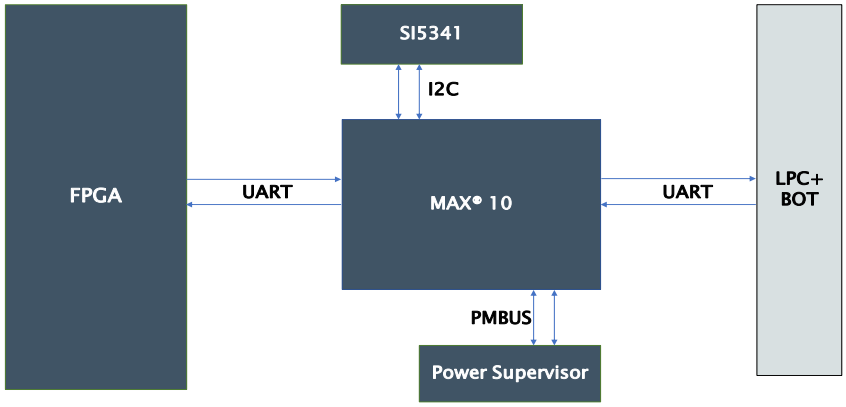



*Figure 34 : Synoptic*

The following block diagram describes the internal architecture of the MAX® 10. The different HDL modules are detailed in the rest of the document.

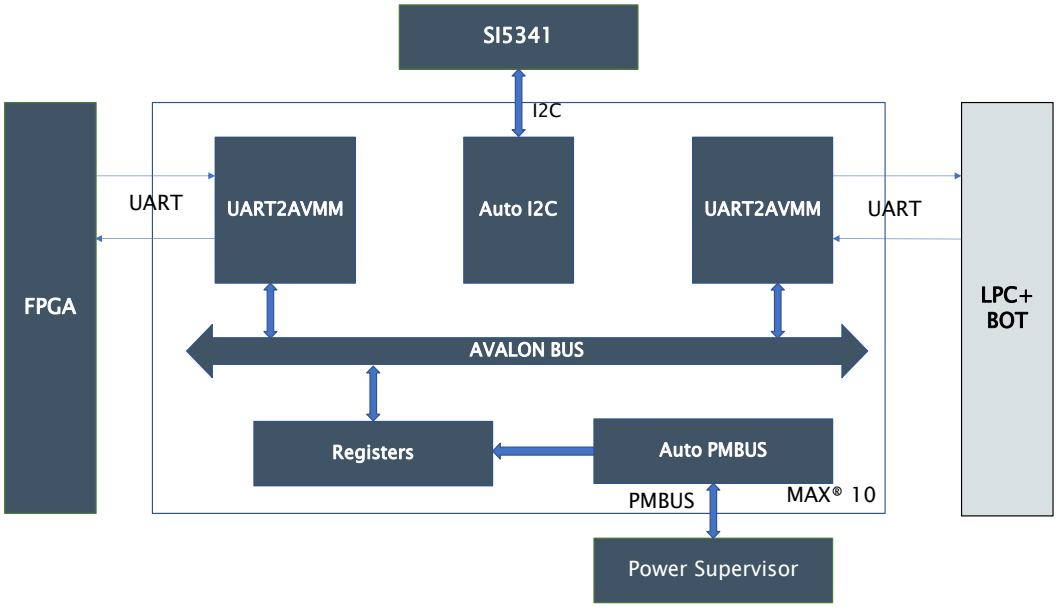



*Figure 35 : MAX® 10 block diagram*

#### 7.14.1.1 Reset sequence

The MAX® 10 is responsible for the reset of the component of the Achilles board.

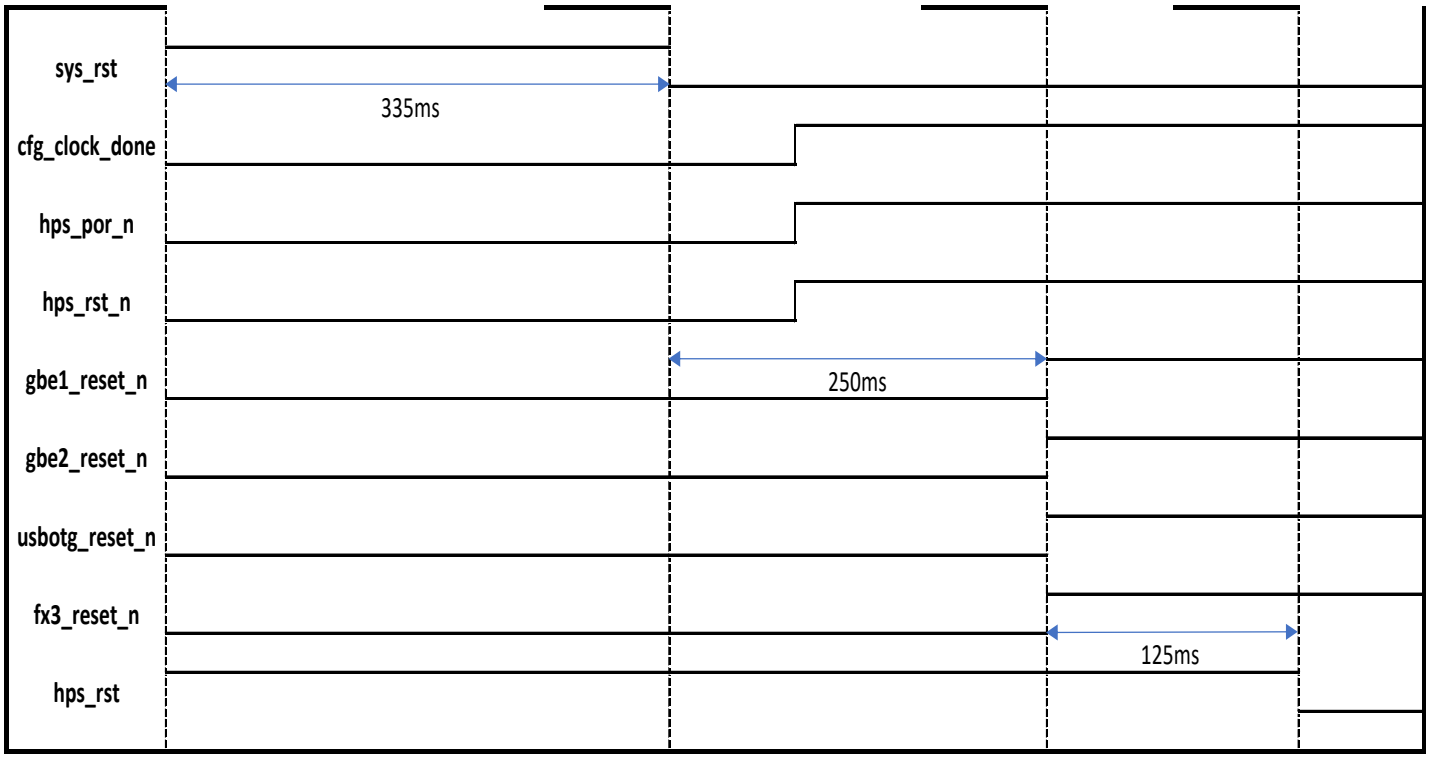



*Figure 36 : Reset sequence time chart*

Reset signals description:

- sys_rst: This is the global system reset for the MAX® 10. It is de-asserted 335ms after the MAX® 10 configuration complete.
- cfg_clock_done: This signal is asserted when the SI5341 is configured.
- hps_por_n: This signal is mapped to the HPS_nPOR. The HPS is maintained in reset until the SI5341 is configured.
- hps_rst_n: This signal is mapped to the HPS_nRST. The HPS is maintained in reset until the SI5341 is configured.
- gbe1_reset_n: This signal is mapped to the reset of the first Ethernet PHY. The reset is de-asserted 250ms after the sys_rst.
- gbe2_reset_n: This signal is mapped to the reset of the second Ethernet PHY. The reset is de-asserted 250ms after the sys_rst.
- usbotg_reset_n: This signal is mapped to the reset of the USBOTG PHY. The reset is de-asserted 250ms after the sys_rst.
- fx3_reset_n: This signal is mapped to the reset of the FX3. The reset is de-asserted 250ms after the sys_rst.
- hps_rst: This signal is mapped to an FPGA IO. The purpose of this signal is to keep the HPS in reset until all its peripherals components have been reset. In our FPGA design, we mapped this signal on the HPS IP warm and cold reset.

#### 7.14.1.2 Auto I2C

This module interface with the clock generator SI5341. It reads data from an ON-CHIP memory and performs I2C access to configure the clock generator. The content of the ON-CHIP memory is generated from a ClockBuilder Pro project.

##### 7.14.1.2.1 How to change the SI5341 configuration

1. SI5341 settings

Run ClockBuilder Pro and build your configuration (./CBP/ folder contains an example project).
Export the settings and save it to config.txt file.
**WARNING**: Verify that "Include summary header" is checked for traceability of the output .txt file.
**WARNING**: Verify that "Include pre- and post-write control register writes" is enabled for correct configuration of SI5341.

2. Generate the .mif

Go to the tools folder.
Edit run_GenMIF.bat to point to the correct .txt file.
If needed, modify the output file name.
Run run_GenMIF.bat.

3. Update MIF

Override the *.mif* file in the Quartus project folder with the new created one.
The steps below generate an updated *.sof* without running the full flow.
This requires that the Quartus database is complete and up to date.
If it is not the case, just run a full compilation process.

4. With full compilation database:

In Quartus, run Processing->Update Memory Initialization File.
In Quartus, run Processing->Start->Start Assembler.
The generated *.sof* now contains the updated *.mif* file.

#### 7.14.1.3 Auto PMBUS

This module interface with the Power Supervisors of the board. It performs PMBUS access each 10ms to read currents and voltages and store the data into a register table.

The main information regarding the PMBUS device are listed below:

PMBUS address:

- LTC2977 A: 0x5C read VOUT/IOUT command: 0xCF
- LTC2977 B: 0x5F read VOUT/IOUT command: 0xCF
- LTM4675: 0x45 read VOUT command: 0x8B read IOUT command: 0x8C

The values read on the PMBUS need to be converted to obtain the real voltage or current value. formulas for the different device are listed below:

LTC2977:

- Voltage value =
- Current value =

LTM4675:

- Voltage value =
- Current value =

The following table list the register offset:

| Reg Offset | Field | Description |
|---|---|---|
| 0x00 | 3V3_VOLTAGE | 3V3 rail voltage sense |
| 0x04 | 3V3_CURRENT | 3V3 rail current sense (Rshunt = 10mΩ) |
| 0x08 | 2V5_VOLTAGE | 2V5 rail voltage sense |
| 0x0C | 2V5_CURRENT | 2V5 rail current sense (Rshunt = 510mΩ) |
| 0x10 | VTT_DDR4_VOLTAGE | VTT DDR4 rail voltage sense |
| 0x14 | VTT_DDR4_CURRENT | VTT DRR4 rail current sense (Not sense) |
| 0x18 | VDDQ_VOLTAGE | VDDQ rail voltage sense |
| 0x1C | VDDQ_CURRENT | VDDQ rail current sense (Rshunt = 100mΩ) |
| 0x20 | 0V9_VOLTAGE | 0V9 rail voltage sense |
| 0x24 | 0V9_CURRENT | 0V9 rail current sense (Rshunt = 5mΩ) |
| 0x28 | 0V95_VOLTAGE | 0V95 rail voltage sense |
| 0x2C | 0V95_CURRENT | 0V95 rail current sense (Rshunt = 10mΩ) |
| 0x30 | 1V2_VOLTAGE | 1V2 rail voltage sense |
| 0x34 | 1V2_CURRENT | 1V2 rail current sense (Rshunt = 30mΩ) |
| 0x38 | 1V8_VOLTAGE | 1V8 rail voltage sense |
| 0x3C | 1V8_CURRENT | 1V8 rail current sense (Rshunt = 10mΩ) |
| 0x40 | 5V_VOLTAGE | 5V rail voltage sense |
| 0x44 | 5V_CURRENT | 5V rail current sense |

*Table 15: Register table*

#### 7.14.1.4 UART2AVMM

This module converts UART signals into Avalon Memory Mapped signals. The UART signals goes through a RS232 module which decodes RS232 protocol into 8 bits Avalon Streaming data. Then the data go into a QSYS system which convert data into packets and into Avalon Memory Mapped signals. The corresponding QSYS system is below:




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

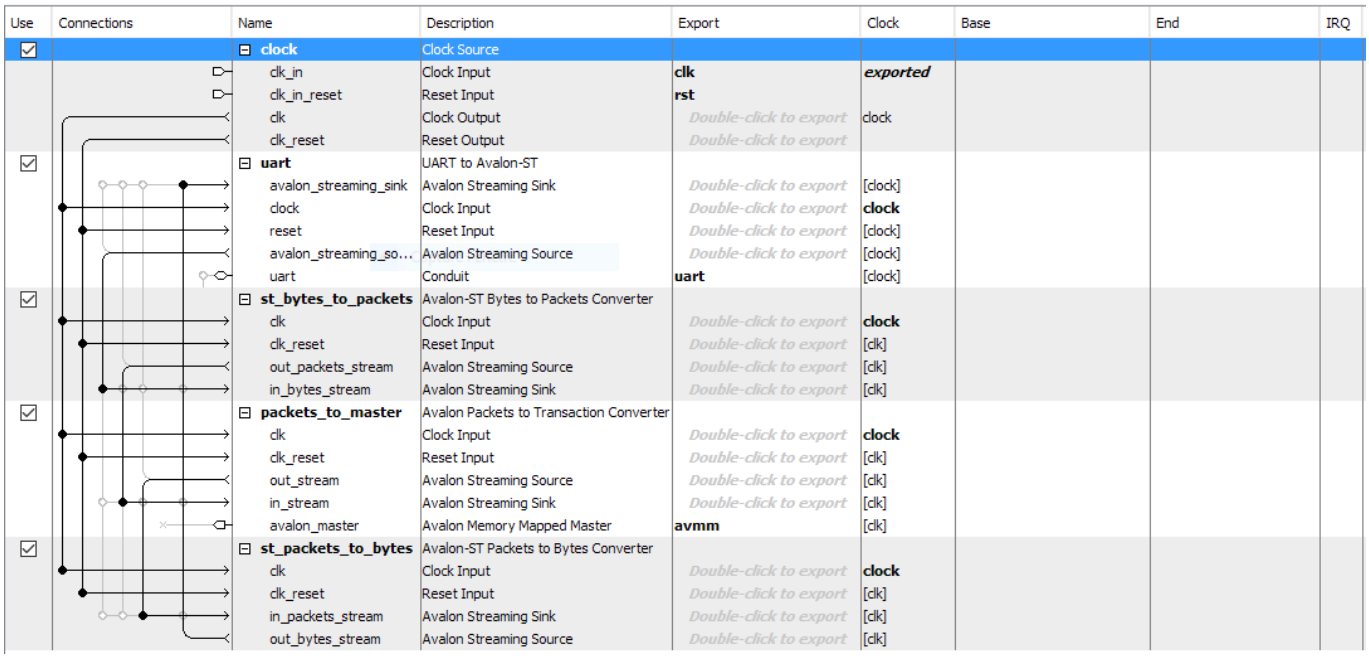



*Figure 37 : UART2AVMM QSYS System*

The following tables describe the protocol used:

| Byte | Field | Description |
|---|---|---|
| 0 | Transaction code | Type of transaction. See Properties of Avalon-ST Interfaces table. |
| 1 | Reserved | Reserved |
| [3:2] | Size | Transaction size in bytes. For write transactions, the size indicates the size of the data field.<br>For read transactions, the size indicates the total number of bytes to read. |
| [7:4] | Address | 32-bit address for the transaction. |
| [n:8] | Data | Transaction data: Data to be written for write transactions. |

*Table 16: Transaction Packet Format*

| Byte | Field | Description |
|---|---|---|
| 0 | Transaction code | Type of transaction. See Properties of Avalon-ST Interfaces table. |
| 1 | Reserved | Reserved |
| [4:2] | Size | Total number of bytes read/written successfully. |

*Table 17: Response Packet Format*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

| Transaction code | Avalon MM Transaction | Description |
|---|---|---|
| 0x00 | Write, non-incrementing address. | Writes data to the given address until the total number of bytes written to the same word address equals to the value specified in the size field. |
| 0x04 | Write, incrementing address. | Writes transaction data starting at the given address. |
| 0x10 | Read, non-incrementing address. | Reads 32 bits of data from the given address until the total number of bytes read from the same address equals to the value specified in the size field. |
| 0x14 | Read, incrementing address. | Reads the number of bytes specified in the size field starting from the given address. |
| 0x7F | No transaction. | No transaction is initiated. You can use this transaction type for testing purposes. Although no transaction is initiated on the Avalon-MM interface, the core still returns a response packet for this transaction code. |

*Table 18: Transaction Supported*

| Operation code | Field | Description |
|---|---|---|
| 0x7D | Escape | The core drops the byte. The next byte is XORed with 0x20. |
| 0x7A | Start of packet | The core drops the byte and marks the next payload byte as the start of a packet by asserting the startofpacket signal on the Avalon-ST source interface. |
| 0x7B | End of packet | The core drops the byte and marks the following byte as the end of a packet by asserting the endofpacket signal on the Avalon-ST source interface. For single beat packets, both the startofpacket and endofpacket signals are asserted in the same clock cycle. |
| 0x7C | Channel number indicator | The core drops the byte and takes the next non-special character as the channel number. |

*Table 19: Operation Avalon-ST Bytes to Packets Converter Core*

For example:

Read @0x20:

| Channel | # | SOP | Read | Reserved | Size | Address (3:1) | EOP | Address (0) |
|---|---|---|---|---|---|---|---|---|
| 7C | 00 | 7A | 14 | 00 | 00 08 | 00 00 00 | 7B | 20 |

Write 0XFF00AA55 @0x20:

| Channel | # | SOP | Read | Reserved | Size | Address (3:0) | Data (3:1) | EOP | Data (0) |
|---|---|---|---|---|---|---|---|---|---|
| 7C | 00 | 7A | 04 | 00 | 00 04 | 00 00 00 20 | 55 AA 00 | 7B | FF |

# 8. HPS DEVELOPMENT

## 8.1 Tools

HPS development is a done on a host PC running Ubuntu 16.04 LTS. This is the Linux distribution recommended by REFLEX CES to minimize compilation issues due to library dependencies. Other distributions may be used to build HPS software, but at the risk for customers to have to solve by themselves some specific issues that REFLEX CES has never been confronted with.

To build and debug HPS software, you need to install the suite of Altera design tools, Altera SoC Embedded Design Suite (SoC EDS PRO, v18.1.0.222). This suite can be download at the following address: http://dl.altera.com/soceds/.

It includes the following tools:

- Quartus Prime Pro 18.1 Programmer and associated tools
- ARM IDE Development Studio 5 (DS-5) – not needed to program binaries but used for hardware and software debug through JTAG probes like USB Blaster or DSTREAM. You should install this tool, which is an Eclipse plugin, during SoC EDS installation.
- bsp-editor to create U-Boot configuration source files, a device tree compiler to create the device tree binary, USB Blaster driver, FTDI driver, and others.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 8.2 HPS Peripherals

### 8.2.1 Supported Features

The following list contains the main modules of the HPS:

- MPU subsystem featuring a dual ARM Cortex-A9 MPCore processors with up to 1.5 GHz CPU operation per core.
- System interconnect that includes three memory-mapped interfaces between the HPS and FPGA:
- HPS-to-FPGA port – 32, 64, or 128 bits wide
- Lightweight HPS-to-FPGA port – 32 bits wide
- FPGA-to-HPS port – 32, 64, or 128 bits wide
- General-purpose direct memory access (DMA) controller
- Security manager
- SECDED supported peripheral memories
- Three Ethernet media access controllers (EMACs)
- Two USB 2.0 on-the-go (OTG) controllers
- NAND flash controller
- Quad serial peripheral interface (QSPI) flash controller
- Secure digital/multimedia card (SD/MMC) controller
- Two serial peripheral interface (SPI) master controllers
- Two SPI slave controllers
- Five inter-integrated circuit (I2C) controllers (1)
- 256 KB on-chip RAM
- 128 KB on-chip boot ROM
- Two UARTs
- Four system timers
- Two watchdog timers
- Three general-purpose I/O (GPIO) interfaces
- ARM Coresight™ debug components:
    - Debug access port (DAP)
    - Trace port interface unit (TPIU)
    - System trace macrocell (STM)
    - Program trace macrocell (PTM)
    - Embedded trace router (ETR)
    - Embedded cross trigger (ECT)
- System manager
- Clock manager
- Reset manager
- FPGA manager

The block diagram below illustrates the HPS:

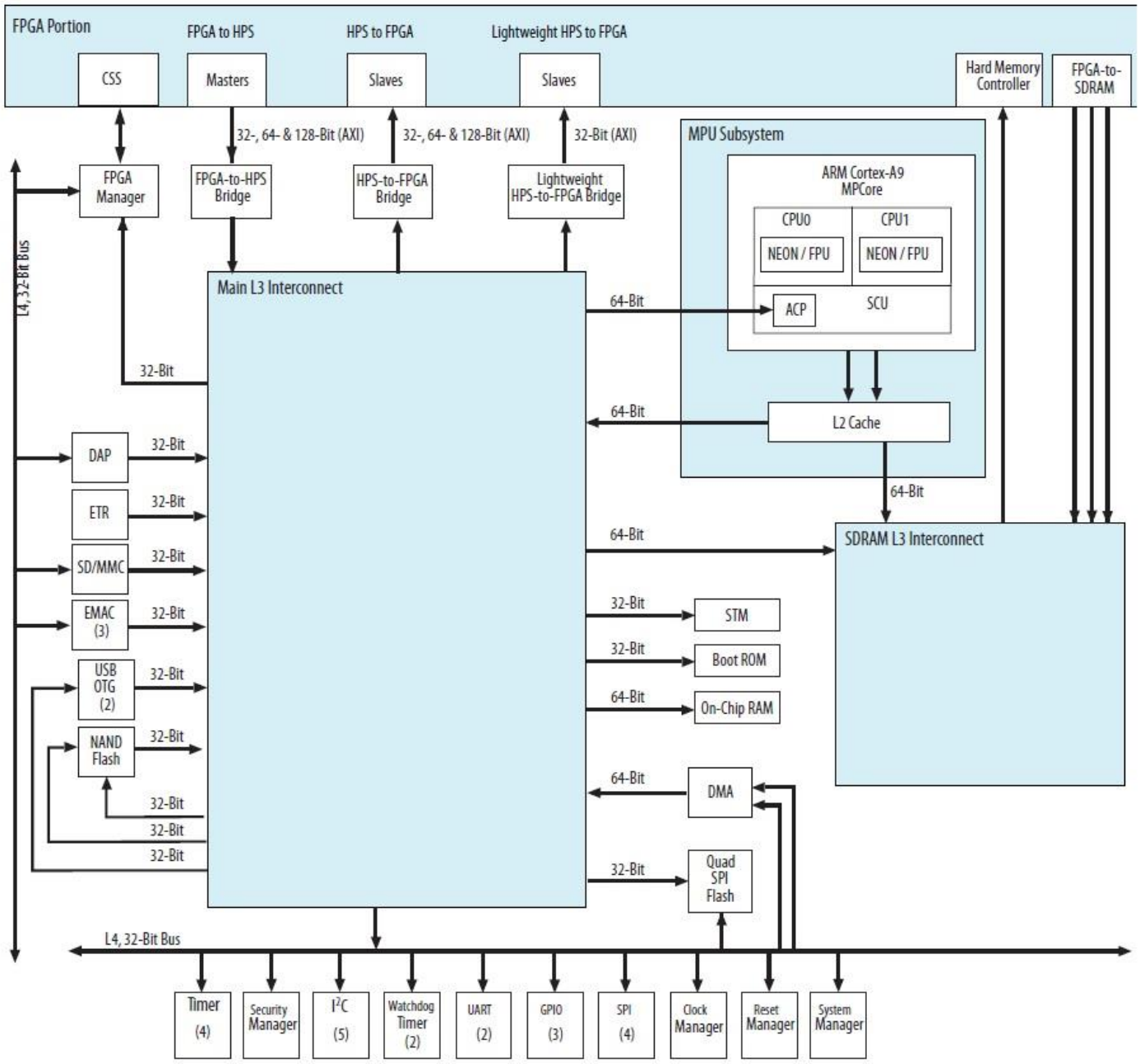



*Figure 38: HPS Block Diagram*

### 8.2.2 Achilles DevKit HPS Peripherals

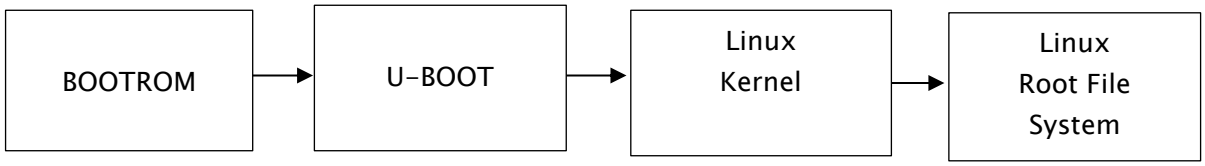
The table below describes the peripherals that HPS supports on the Achilles DevKit.

| Peripheral | Description |
|---|---|
| **DDR4** | 32-bit DDR4 memory, composed of two 16-bit MT40A1G16WBU-083E components, with a capacity of 4GB. Only 3GB is currently supported, however. |
| **eMMC** | An eMMC is supported though the SD/MMC controller. This is a MTFC32GJDDQ-4M IT component from Micron. |
| **UART** | A UART is connected to the HPS through the LPC+ HPS BOT connector also referred to as the Bottom-Right connector |
| **Ethernet** | Two Gigabit Ethernet ports. These PHY components are the KSZ9031RNXCA from Micrel. |
| **USB 2.0 OTG** | USB 2.0 OTG interface. The ULPI PHY component is the USB3320C-EZK from Microchip. An important limitation is that **it can only be used in device mode** i.e. you can only use USB devices with this interface. |
| **LEDs** | 2 LEDs, green and red, connected to dedicated GPIO. |
| **Temperature Sensor** | A temperature sensor connected to its I2C bus (address 0x48). This temperature sensor is a TMP102AIDRLR from Texas Instruments. |
| **Real-Time Clock** | A real-time clock (RTC) connected to its I2C bus (address 0x51). This real-time clock is a PCF8563TS/5 from NXP. |
| **EEPROM** | When the Achilles module is plugged into the Starter Board, the HPS also supports an EEPROM 24LC024-I/SN from Microchip connected to its I2C bus (address 0x54). |

*Table 20: Supported peripherals by HPS on the Achilles DevKit*

## 8.3 HPS Software Architecture

The following software blocks are used when the HPS powers-up to become ready to use:




### 8.3.1 BootROM

When power is switched on, the processor executes the 128 KB BootROM code that resides inside on-chip ROM. The main role of the BootROM is to initialize all required hardware components to boot up the next stage boot software, the U-Boot bootloader.

The BootROM can fetch the bootloader binary from several storage devices based on the BSEL pins. On the Achilles DevKit, this is the eMMC that is used to store all the software blocks.

### 8.3.2 U-Boot

The bootloader used in the DevKit is U-Boot, an open source bootloader used in many embedded devices.

The main functions of U-Boot are:

- Setting up the OS environment
- Fetching the boot image from eMMC or Ethernet through TFTP
- Storing the boot image to SDRAM and passing control over to this image
- Providing a console that can be used for user operations, such as modifying boot arguments or accessing supported devices.

### 8.3.3 Linux Kernel

The Operating System running on the HPS is Linux, so the boot image loaded by U-boot is the kernel image. It is responsible for supporting all the hardware of the Achilles DevKit handled by the HPS. The version used is a customized 4.1.22.

### 8.3.4 Linux Root File Systems

The root file system mounted by the Linux kernel on the HPS is based on the Linaro-developer-armhf release based on Debian stretch. It has been customized to add SSH support, network setup and HPS test scripts and programs.

Another specific root file system is used during the factory boot flow to partition the eMMC (see next chapter for boot flow details). This is a RAM file system integrated to the kernel image. It is built with Buildroot and based on Busybox.

## 8.4 HPS Boot Flow

This section introduces the boot flow used in factory mode and default mode.

The default boot flow is created when the eMMC has been partitioned and software binaries have been copied into it. This boot flow is used in 99% of cases.

The factory boot flow is used when the eMMC is empty. REFLEX CES use this mode to program the Achilles board.

This section describes the boot flow. Later sections explain how to build the software elements used in both default and factory modes.

### 8.4.1 Achilles DevKit Default Boot Flow

In the default boot flow, the HPS boot and FPGA configuration occur separately. The default boot flow is illustrated below:

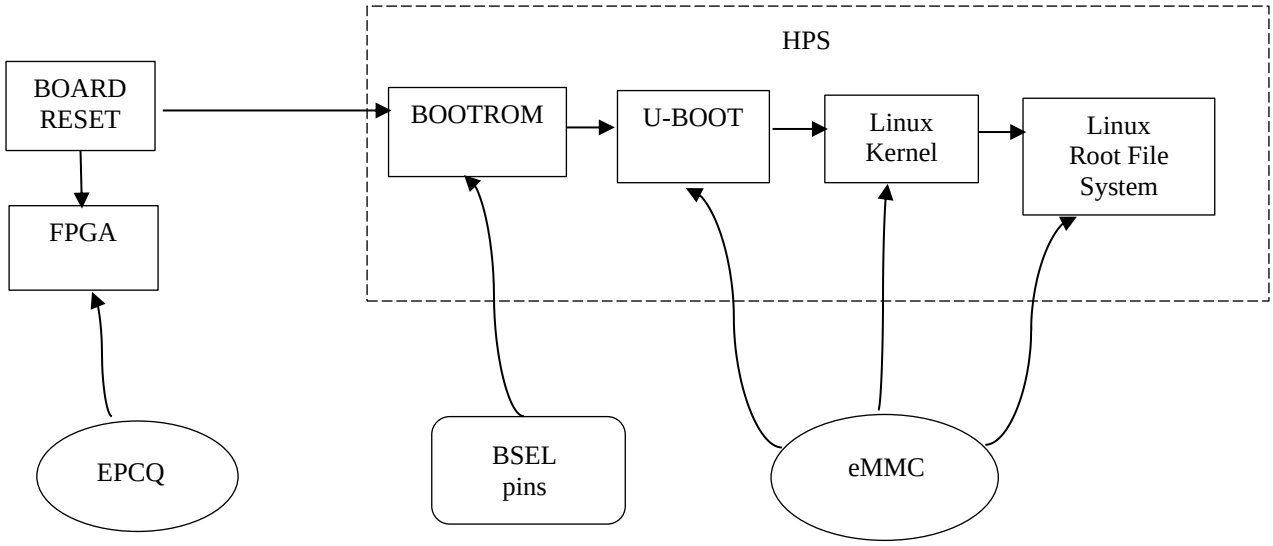



*Figure 39: Default Boot Flow*

1. At power-up, the FPGA takes its configuration from EPCQL while the BootROM starts its execution on CPU0 (CPU1 is held in reset while boot ROM executes on CPU0).

2. After initializing the HPS, the BootROM determines the device where the bootloader is stored using the BSEL pins. As previously mentioned, you can store the bootloader in several different storage devices. The Achilles configuration uses the eMMC device. U-Boot is stored in a specific partition with type=0xA2. This is required by the BootROM, which will identify it from the MBR (Master Boot Record) and loads U-Boot from the beginning. This boot choice is also fixed by the FPGA hardware. For eMMC configuration, the BSEL [2:0] value should be set to 0x5. Any other BSEL values will prevent HPS from booting. See Section 8.8.1: BSEL Configuration8.8.1 for more information.

3. The BootROM then copies U-Boot into on-chip RAM and executes it.

4. U-boot continues the HPS initialization, then loads the Linux kernel and its device tree from another dedicated eMMC partition.

5. At the end of its execution, the kernel mounts the root file system stored on a third eMMC partition.

For more information about how to build the U-boot, Linux kernel and device tree, and root file system used here, see Section 8.7.1: Shell environment mandatory requirements
Before trying to build HPS software from a terminal command, some mandatory steps are required:

- The toolchain provided in the USB key Tools folder must be extracted on the development PC.
- Some variables must be exported in the shell, either by explicitly setting them in the shell or through environment files:
  - export CROSS_COMPILE="<path_to_toolchain_exctracted/bin/arm-linux-gnueabihf->".
    For example *export CROSS_COMPILE="/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-"*
  - *export ARCH=arm*
  - export SOCEDS_ROOT_PATH="<path_to_your_SoCEDC_installation>".
    For example *export SOCEDS_ROOT_PATH="/opt/intelFPGA_pro/18.1/embedded"*

Automatic Build.

### 8.4.2 Achilles Devkit Factory Boot Flow

The factory boot flow is used for the very first boot when the eMMC has not yet been partitioned or when the eMMC has been corrupted and need to be restored.

Factory boot flow boots a root file system and uses a partitioning script based on Linux fdisk utility to create the eMMC partitions. Because the U-Boot cannot be loaded from eMMC, this flow uses the specific HDL design 'RefDesign_HPS_boot_from_FPGA' to make the HPS boot from the FPGA. This design stores the U-boot in hex format in an on-chip RAM.

The factory boot flow is illustrated below:

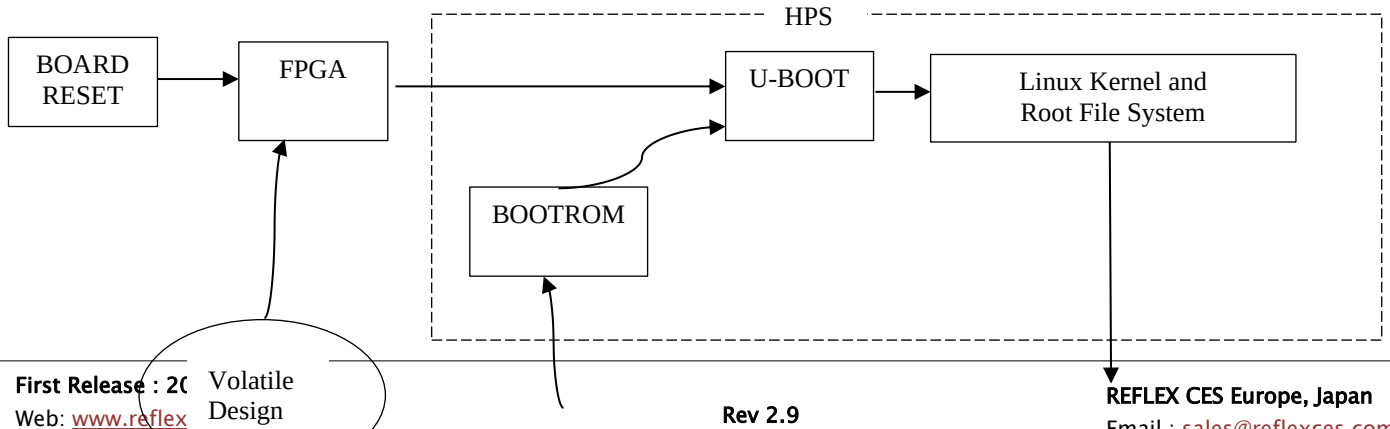



*Figure 40: Factory Boot Flow*

1. The HDL design loads the U-Boot in OCRAM.
2. The U-Boot then downloads a specific kernel that includes a RAM root file system using TFTP and boots the kernel.
3. On the TFTP server this kernel image must be named "zImage". See Section 8.7: Build Flow for a description of how to build this kernel.
4. When the root file system is loaded, the partitioning script located in /root can be executed to create the eMMC partitions. Depending on the way the kernel image has been built, the script can be automatically executed. This is not the case for the zImage file provided in the USB key to let customers customize the script according to their network configuration.
5. This script downloads by TFTP all the final software binaries (built with the target "all" using the Autobuild – see Section 8.7) and copies them into the right partition:
    - U-Boot must be named "uboot.bin" on the server
    - Linux kernel must be named "zImage.default" on the server
    - Linux device tree must be named "linuxDT.dtb" on the server
    - Root file system must be named "rootfs.tar.gz" on the server.

For more information about how to build the U-boot in hex format and the Linux kernel that includes a RAM-based root file system used here, see the "Factory" section of Section 8.7.1: Shell environment mandatory requirements
Before trying to build HPS software from a terminal command, some mandatory steps are required:

- The toolchain provided in the USB key Tools folder must be extracted on the development PC.
- Some variables must be exported in the shell, either by explicitly setting them in the shell or through environment files:
    - export CROSS_COMPILE="<path_to_toolchain_exctracted/bin/arm-linux-gnueabihf->".
      For example *export CROSS_COMPILE="/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-"*
    - *export ARCH=arm*
    - export SOCEDS_ROOT_PATH="<path_to_your_SoCEDC_installation>".
      For example *export SOCEDS_ROOT_PATH="/opt/intelFPGA_pro/18.1/embedded"*

Automatic Build.

## 8.5 eMMC Layout

When the eMMC has been partitioned, the eMMC layout is as follows:

| | | |
|---|---|---|
| | Partition4 (VFAT)<br>Empty | Remaining space |
| | Partition 3 (EXT3)<br>Root File System | 4 GB |
| | Partition 2 (VFAT)<br>Linux kernel<br>Linux device tree | 50 MB |
| | Partition 1 (RAW)<br>U-Boot | 10 MB |
| | U-Boot environment | 4 KB |
| 0x0000 0000 | Master Boot Record | 512 bytes |




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 8.6 HPS Folder Contents

This section describes the contents of the HPS folder on the USB key. The HPS folder content may be provided in a tarball archive, in such case just extract all files in a HPS folder at the same tree level.

### 8.6.1 Autobuild

This directory contains the *build.sh* shell script and several directories used by this script to automatically build all the software blocks. The script prompts the user for input at certain build steps.
It contains the following files:

| File | Description |
|---|---|
| **Buildroot-inputs** | Contains all Buildroot Achilles-specific files that will be copied by the script before building:<br>• Buildroot configuration files<br>• U-boot source code patch for default and factory mode<br>• Kernel configuration file<br>• Kernel device tree source file<br>• Kernel source patch<br>• Root file system overlay files<br>• A script file to concatenate U-Boot binary and its device tree |
| **rootfs** | Contains the Linaro-nano root file system archive based on Debian Stretch and customized for test purposes. |
| **uboot** | Contains U-Boot default and factory device tree source files. |
| **build.sh** | Autobuild script used to generate software binaries for both boot flows. |

*Table 21: Script directories*

### 8.6.2 Factory

This directory contains the kernel image with Busybox RAM file system used for eMMC partitioning. In this image, the partitioning script is located in /root and is not automatically executed.

A README file gives details on default TFTP server configuration expected by the partitioning script.

### 8.6.3 Kernel

This directory contains all binaries and source files relative to the Linux kernel.

| File | Description |
|---|---|
| **Bin** | Contains the following files:<br>• *linuxDT*: Linux device tree<br>• *zImage*: Linux kernel image<br>• *README.txt*: information file on directory content. |
| **Src** | Contains the following files:<br>• *socfpga_rxc_defconfig*: kernel config file specific to Achilles DevKit<br>• *kernel.patch*: patch to customize original git repository source files. The git repository used is git://github.com/altera-opensource/linux-socfpga.git, socfpga-4.1.22-ltsi branch. The commit this patch is based on, is 9689ce4f759ad0f13f6911d1f1309384f2b05f91 corresponding to the tag 'rel_socfpga-4.1.22-ltsi_16.10.02_pr'.<br>This repository is now archived, and the archive can be obtained from this link: https://github.com/altera-opensource/linux-socfpga/archive/9689ce4f759ad0f13f6911d1f1309384f2b05f91.tar.gz<br>• The patch adds support for REFLEX CES configuration of the board.<br>• *README.txt*: information file indicating how to retrieve Linux kernel source for the Achilles board.<br>• *devicetree:* directory containing Linux device tree source files and xml files to re-generate this source file:<br>  • *linuxDT.dts*: device tree source file, used to generate the device tree binary<br>  • *hps.sopcinfo*: hardware design output file from QSYS, used to generate device tree source files<br>  • *README.txt*: information to generate the device tree binary from the source file.<br>  • *hps_a10_common_board_info.xml*: board info file<br>  • *ghrd_10a066h4f34e3sg_board_info.xml*: board info file<br>  • *hps_a10_devkit_board_info.xml*: board info file<br>• *linux_download.sh*: script that downloads and applies the patch to the kernel. |

### 8.6.4 Rootfs

This directory contains a README file providing root file system archive location in HPS folder.

### 8.6.5 Test

This directory contains source code and scripts for several tools that enable the testing of certain HPS components:

- *benchmark*: folder source code for coremark and stream benchmark tools.
- *hps-good*: folder source code for hpsgood utility (valid only with GUI design firmware, used to notify that HPS has booted successfully).
- *memtester*: source code and script file for memtest application.

- *eepromtest.sh*: script to test the EEPROM on the Achilles Starter Board.
- *ledtest.sh*: script testing the two HPS LEDs.
- *rxc_monitor*: daemon to push monitoring data to the graphical user interface (Achilles DevKit only).
- *rtctest.sh*: script to test the real-time clock.
- *tempsensortest.sh*: script to test the temperature sensor.
- *usbtest.sh*: script to test USB OTG.

### 8.6.6 Tools

This directory contains the Linaro toolchain used to build U-Boot and Kernel images without using autobuild script.

### 8.6.7 U-Boot

This directory contains all binaries and source files relative to the U-Boot bootloader.

| File | Description |
|---|---|
| **Bin** | Contains the following files:<br>• *uboot_w_dtb-mkpimage.bin*: U-Boot binary and device tree concatenated<br>• *README.txt*: information file containing folder contents. |
| **Src** | Contains the following files:<br>• *hps_isw_handoff*: directory containing hardware design output files from QSYS. This is the input directory for Altera bsp-editor tool. It enables the generation of configuration source files and the makefile to build U-Boot bootloader.<br>• *uboot.patch*: U-Boot patch used to customize the original git repository source files for default boot flow. The git repository used is git://github.com/altera-opensource/u-boot-socfpga.git, socfpga_v2014.10_Arria10_bringup branch.<br>• *README.txt*: information file describing the current folder.<br>• *Makefile*: file that should be used instead of the file generated by the Altera bsp-editor. It enables the use of U-Boot source files from the Altera git repository instead of the pre-installed source archive included with the SoC EDS installation.<br>• *bsp-editor.sh*: script that executes the Intel PSG bsp-editor tool.<br>• *git_download.sh*: script that downloads and applies patch to Uboot. |

### 8.6.8 ReleaseNotes.txt

This file contains the Release Notes for the Achilles DevKit.

## 8.7 Build Flow

All this build flow chapter is available to **Linux host system only.**

This section describes how to build the different software components.

The Achilles DevKit is based on the Arria® 10 SoC, which is also used on the Altera Arria® 10 SoC Board. The Altera build flow is illustrated in the following diagram:

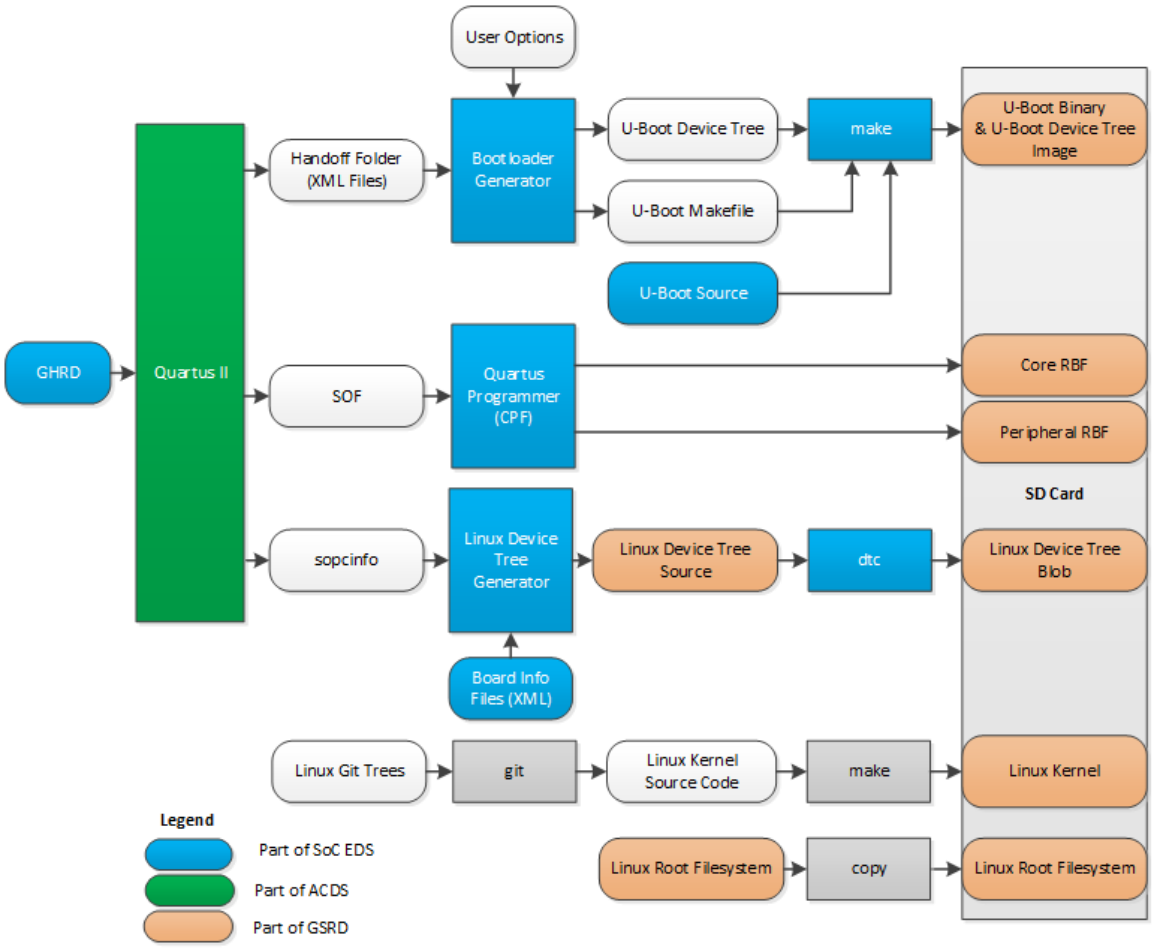



*Figure 41: Global Build Flow*

The key characteristics of this build system are:

- It uses Quartus output files as input files: handoff directory, SOF file and sopcinfo file.
- It uses several tools provide by the SoC EDS: the bootloader generator (named bsp-editor), the Linux device tree generator, and the Linux device tree compiler (dtc).
- It uses the archive file included with the SoC EDS installation for U-Boot source code, as a git repository for Linux kernel source code and as a Yocto build for the root file system.

*NOTE: All the documentation for Altera's SoC boards (based on Cyclone V, Arria® V and Arria® 10) can be found on the rocketboards website: http://rocketboards.org/foswiki/view/Documentation/WebHome.*

The Achilles DevKit **does not** follow the same procedure to build Achilles software. In the Achilles build flow, the U-Boot and the Linux kernel can be built manually using customized Altera tools, or automatically using Buildroot. We highly recommend an automatic build using Buildroot.
The following sections describe the automatic build, and then explain how to manually build the U-Boot and the Linux kernel for the default boot flow.

### 8.7.1 Shell environment mandatory requirements

Before trying to build HPS software from a terminal command, some mandatory steps are required:

- The toolchain provided in the USB key Tools folder must be extracted on the development PC.
- Some variables must be exported in the shell, either by explicitly setting them in the shell or through environment files:
  - export CROSS_COMPILE="<path_to_toolchain_exctracted/bin/arm-linux-gnueabihf->".
    For example *export CROSS_COMPILE="/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-"*
  - *export ARCH=arm*
  - export SOCEDS_ROOT_PATH="<path_to_your_SoCEDC_installation>".
    For example *export SOCEDS_ROOT_PATH="/opt/intelFPGA_pro/18.1/embedded"*

### 8.7.2 Automatic Build

The simplest way to build Achilles software is to use the *build.sh* script provided in the *Autobuild* directory of the USB key and available on **Linux host system only**.

To perform the automatic build, you need to install some packages:

```
apt-get install sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget git device-tree-compiler
```

*NOTE: Several directories will be created during the build, so you should not execute the build from the USB key. Instead, copy the directory <USB_Key/HPS/Autobuild> to your PC.*

To execute the *build.sh* script, open a console, move to the script directory, and launch the command *./build.sh <target>*.
The different possible targets are described below.

| Target | Description |
|---|---|
| **help** | Displays the script help. This is the default target if no argument is passed to the script. |
| **all** | **This target builds all the software elements for the default boot flow**. It builds U-Boot, Linux device tree and Linux kernel. It copies the Linaro root file system alongside other binaries.<br><br>During this build:<br>1) You are asked if you want to use the default U-Boot device tree. The default choice is yes:<br>• If yes, then the device tree source file used is *Autobuild/uboot/UbootDT-default.dts*.<br>• If no, then the scripts check if a previous installation exists. You are asked whether you want to remove or use the existing installation. If you want to remove the existing installation, then you are asked to enter the path to its handoff directory, generated by QSYS, to copy it to the local *Autobuild/uboot* directory. The *bsp-editor* tool is then launched to generate the U-Boot device tree source.<br><br>2) A *downloads* directory is created for all future downloads. Buildroot source code is downloaded in this directory, then extracted to the *Autobuild* directory. It creates the *buildroot-2015.08* directory.<br><br>3) The Buildroot configuration file *Autobuild/buildroot-inputs/rxc_achilles_default_defconfig* is copied to *buildroot-2015.08/configs* folder and renamed *rxc_achilles_defconfig*. Other *Autobuild/buildroot-inputs* files are copied to a new directory called *achilles* in *buildroot-2015.08/board*. These files are:<br>• *post-image.sh*: a script that builds the U-Boot device tree binary.<br>• *kernel/patches/kernel.patch*: a Linux kernel source code patch.<br>• *kernel/linuxDT.dts*: a Linux kernel device tree source.<br>• *kernel/socfpga_rxc_defconfig*: a Linux kernel configuration.<br>• *uboot/uboot-default.patch*: U-Boot source code patch.<br><br>4) The *make rxc_achilles_defconfig* is executed to apply Achilles configuration. |

| Target | Description |
|---|---|
| | 5) The *make* command is executed from *buildroot-2015.08.* This command builds the U-Boot (*uboot_w_dtb_mkpimage.bin*), the Linux device tree (*linuxDT.dtb*) and the Linux kernel (*zImage*) in *buildroot-2015.08/output/images.*<br><br>6) The linaro-nano root file system *custom_linaro-vivid-nano-20151515-714.tar.gz* is copied to *buildroot-2015.08/output/images.*<br><br>***IMPORTANT NOTE:***<br>*For the factory boot flow, the binaries built here need to be uploaded to a TFTP server. Be careful to change some binaries names once uploaded because the partitioning script expects some specific names:*<br>• *uboot_w_dtb_mkpimage.bin must be renamed to uboot.bin*<br>• *linuxDT.dtb should remain unchanged.*<br>• *zImage must be renamed zImage.default*<br>• *custom_linaro-vivid-nano-20151515-714.tar.gz must be renamed rootfs.tar.gz* |
| **factory [auto/manual]** | **This target builds all the software elements for the factory boot flow**:<br>1) U-Boot in hex format that is then embedded in the HDL RefDesign_HPS_boot_from_FPGA design.<br>2) Linux kernel including the RAM root file system based on Busybox to partition the eMMC. It must be then copied to the TFTP server.<br><br>This target takes additional optional argument taking two possible values to control the partitioning script execution:<br>• auto: means that the partition script is automatically executed at boot.<br>• manual: means that the partition script must be manually executed after boot.<br>If only the "factory" argument is given as argument to the build.sh script, then the argument "auto" is automatically assumed. As the partitioning script expects a specific network configuration, we highly recommend to customers to execute the command *buid.sh factory manual* to be able to customize the script according to their network environment before it be executed.<br><br>During this build:<br>3) You are asked if you want to use the default U-Boot device tree. The default choice is yes:<br>a If yes, then the device tree source file used is *Autobuild/uboot/UbootDT-default.dts*.<br>b If no, then the scripts check if a previous installation exists. You are asked whether you want to remove or use the existing installation. If you want to remove the existing installation, then you are asked to enter the path to its handoff directory, generated by QSYS, to copy it to the local |

| Target | Description |
|---|---|
| | *Autobuild/uboot* directory. The *bsp-editor* tool is then launched to generate the U-Boot device tree source.<br><br>4) A *downloads* directory is created for all future downloads. Buildroot source code is downloaded in this directory then extracted to the *Autobuild* directory. It creates the *buildroot-2015.08* directory.<br><br>5) The Buildroot configuration file *Autobuild/buildroot-inputs/rxc_achilles_factory_defconfig* is copied to *buildroot-2015.08/configs* and renamed *rxc_achilles_defconfig*. Other *Autobuild/buildroot-inputs* files are copied to a new directory called *achilles* in *buildroot-2015.08/board*. These files are:<br>a *post-image.sh*: a script that builds the U-Boot device tree binary.<br>b *kernel/patches/kernel.patch*: a Linux kernel source code patch.<br>c *kernel/linuxDT.dts*: a Linux kernel device tree source.<br>d *kernel/socfpga_rxc_defconfig*: a Linux kernel configuration.<br>e *rootfs-overlay-factory/**: this file overwrites or adds new files to the root file system, mainly to automatically create the eMMC partitions at boot using *partitionning.sh*.<br>f *uboot/uboot-factory.patch*: U-Boot source code patch.<br><br>6) The *make rxc_achilles_defconfig* is executed to apply Achilles configuration.<br><br>7) The *make* command is launched from *buildroot-2015.08.* This command builds the U-Boot (*hpsBootTarget.hex*) and the Linux kernel that includes the RAM root file system based on Busybox (*zImage*) in *buildroot-2015.08/output/images.*<br><br>The generated U-Boot in hex format is used by the HDL refDesign_HPS_boot_from_FPGA design and is loaded into the FPGA on-chip RAM.<br><br>The generated kernel image with RAM file system *zImage* must be copied into the TFTP server (keeping the same name). It is loaded by the U-Boot from refDesign_HPS_boot_from_FPGA design. The partitioning script included in the embedded root file system will retrieve from the TFTP server the final binaries built by the *./build.sh all* command.<br><br>***IMPORTANT NOTE:***<br>*Before starting the factory boot flow, the software binaries for the default boot flow must first be built and copied in a TFTP server as they are retrieved from the TFTP server and copied into the correct eMMC partition by the partitioning script. Do not forget to respect the binaries naming rule in the TFTP server explained in the above note.* |

| Target | Description |
|---|---|
| **clean** | The *make clean* command is launched from *buildroot-2015.08.* It removes the *buildroot-2015.08/board/achilles* and *buildroot-2015.08/output/images* folders, as well as others. |
| **distclean** | This command restores the *Autobuild* directory to its original state, except for the *downloads* folder that is not removed. The command prompts you for confirmation before deleting files. |
| **rebuild** | This command launches a distclean and then builds all targets. |

### 8.7.3 Root File Systems

- The linaro-nano root file system is provided as a compressed archive and cannot be rebuilt. It has been customized to add a SSH server (see Section 8.8.3: Ethernet Connection for connection information) and different test scripts and programs. It is mounted with read-only access. To enable write access; after the board boots, enter the command *mount -o remount,rw /* in the terminal.

- The Busybox-based root file system built during the automatic factory build contains most of the classical Linux commands. Some specific commands have also been added, such as the Dropbear package to add an SSH server and to test certain components. You can add other packages by executing the *make menuconfig* command from the *buildroot-2015.08* directory and selecting the package from the Target packages menu.
  Busybox can also be configured by executing the *make busybox-menuconfig* command. A root file system overlay enables the customization of file system content.

### 8.7.4 U-Boot Unitary Build

All this chapter is available on **Linux host system only**.

#### 8.7.4.1 Altera DevKit vs Achilles DevKit Build Flow

The Altera U-Boot build flow is illustrated below:

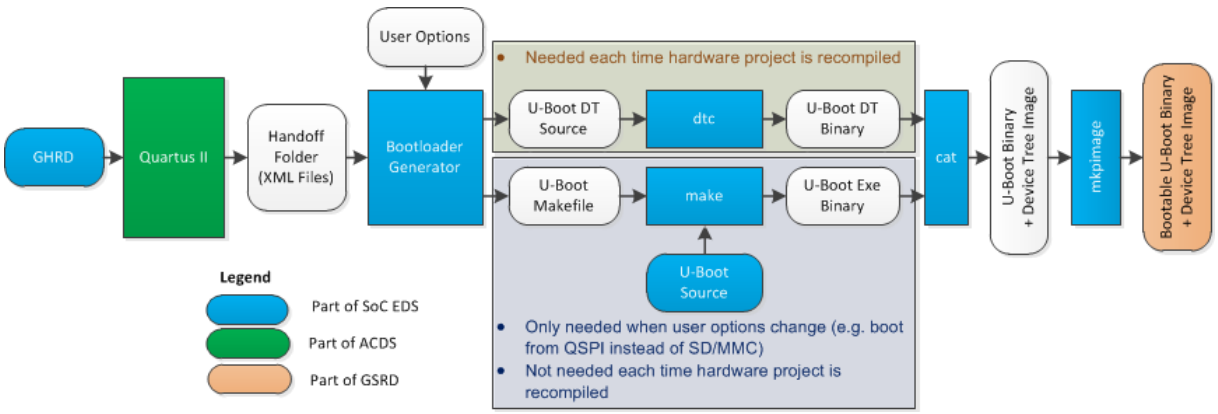



*Figure 42: U-Boot Build Flow*

Achilles uses a similar build flow with one important exception.
In the Altera flow, the U-Boot Makefile generated by the Bootloader Generator uses as U-Boot source code the archive provided by the SoC EDS installation and located in <SoCEDS_Root_Path>/intelFPGA/18.1/embedded/host_tools/altera/bootloaders/u-boot/uboot-socfpga.tar.gz. The makefile extracts the archive, then compiles the source code. This means that the source code cannot be modified.

This **does not** work for the Achilles DevKit, however, as the U-boot source code needs to be modified to incorporate the custom Achilles DevKit hardware. To avoid this issue, Achilles uses a git repository provided by Altera to store the U-Boot source code, as is done for the Linux Kernel source. This means that the U-Boot source code can be modified to incorporate Achilles hardware. It can then be compiled using a custom Makefile.

#### 8.7.4.2 Handoff Directory

The U-Boot build input is the handoff directory located on the USB key: <USB_Key>/HPS/U-Boot/src/hps_isw_handoff.
It contains an *emif.xml* file that provides RAM timings for the U-BOOT DDR configuration and an *hps.xml* file that contains the results of the HPS instantiation in QSYS.
This *hps.xml* file provides the I/O pin muxing and indicates which I/Os are shared between the FPGA and which are assigned to the FPGA and HPS. There are 17 pins assigned to the HPS for specific devices. 48 pins are shared with the FPGA and organized in four blocks of 12 pins. Each block can be assigned to either the FPGA or the HPS. The following figure illustrates this I/O configuration:




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

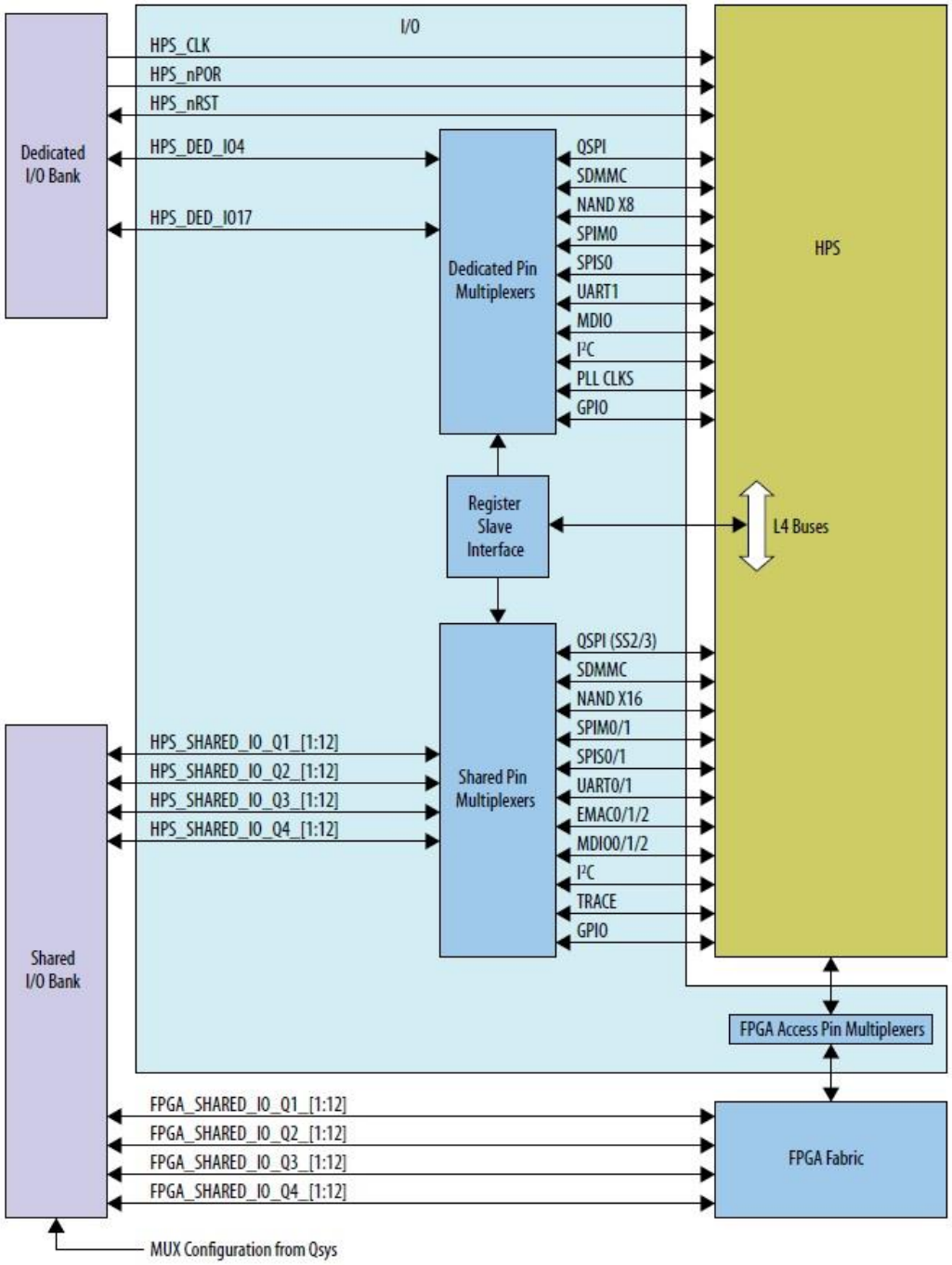



*Figure 43: I/O Configuration*

### 8.7.4.3 Git Repository

The git repository for the U-Boot source code is provided by Altera and can be found on the rocketboards website at the following address: https://github.com/altera-opensource/u-boot-socfpga. It uses the commit ab2181dd766157a74b309d12e0b61c4f3cdc8564.

#### 8.7.4.4 Source Code Modifications for Default Boot Flow

To adapt the Altera U-Boot source code to the Achilles hardware platform for default boot flow, a uboot.patch file has been created. This patch file:

- creates a configuration file for Achilles: configs/socfpga_arria10_rxc_defconfig
- adds DDR4 timings for the Achilles board to: arch/arm/include/asm/arch-socfpga_arria10/sdram.h
- modifies the U-Boot configuration file for Achilles DevKit: include/configs/socfpga_arria10.h
  - adds RXC_CONFIG_BOOT_XXX to select boot device configuration
  - creates several boot variables to enable booting in different configurations
  - adds use of EMAC1 controller.
  - changes SDRAM size (3GB)
  - adds use of UART0 controller.
  - adds NFS support
  - changes eMMC bus width (8 bits)

#### 8.7.4.5 Source Code Modifications for Factory Boot Flow

To adapt the Altera U-Boot source code to the Achilles hardware platform for factory boot flow, a uboot.patch file has been created. This patch file:

- creates a configuration file for Achilles: configs/socfpga_arria10_rxc_defconfig
- adds DDR4 timings for the Achilles board to: arch/arm/include/asm/arch-socfpga_arria10/sdram.h.
- modifies the security/privilege settings in the Arria 10 NOC: board/altera/socfpga_arria10/socfpga_common.c to allow masters to write to the FPGA OCM.
- modifies the U-Boot configuration file for Achilles DevKit: include/configs/socfpga_arria10.h:
  - adds RXC_CONFIG_BOOT_FROM_NET to enable Ethernet boot device configuration
  - adds use of EMAC1 controller.
  - creates several boot variables to enable booting over a network
  - changes SDRAM size (3GB)
  - adds use of UART0 controller.
  - changes eMMC bus width (8 bits)

To adapt this change to your own network configuration, in the socfpga_arria10.h file, edit the variable *serverip* to match the host PC IP address. You should also edit the variable *rxcethboot* to change the Linux network configuration.

Note that the factory boot flow relies on the RefDesign_HPS_boot_from_FPGA HDL firmware.

#### 8.7.4.6 Bsp-editor

The Bsp-editor allows generating the U-boot bootloader. Using the files in the handoff directory, it generates all the files required to build the U-Boot binary and its device tree: the device tree source file, configuration file, Makefile and a useful initialization boot script for ARM DS-5.

#### 8.7.4.7 Building the U-Boot and its Device Tree

The previous sections described the various build components. This section describes how to build a U-Boot image.

0) Be sure that the SOCEDS_ROOT_PATH, CROSS_COMPILE and ARCH variables are already set in your shell environment as explained in previous chapter 8.7.1.

1) Go into the <USB_Key>/HPS/U-Boot/src directory:
*replacing <USB_Key> with the folder you have copied the files to on your PC*

```
cd <USB_Key>/HPS/U-Boot/src
```

2) Download the U-Boot source files from the GitHub repository, and apply the patch using the following script:

```
./git_download.sh
```

3) Set the environment variable UBOOT_SRC_DIR with the path of U-Boot sources previously downloaded:

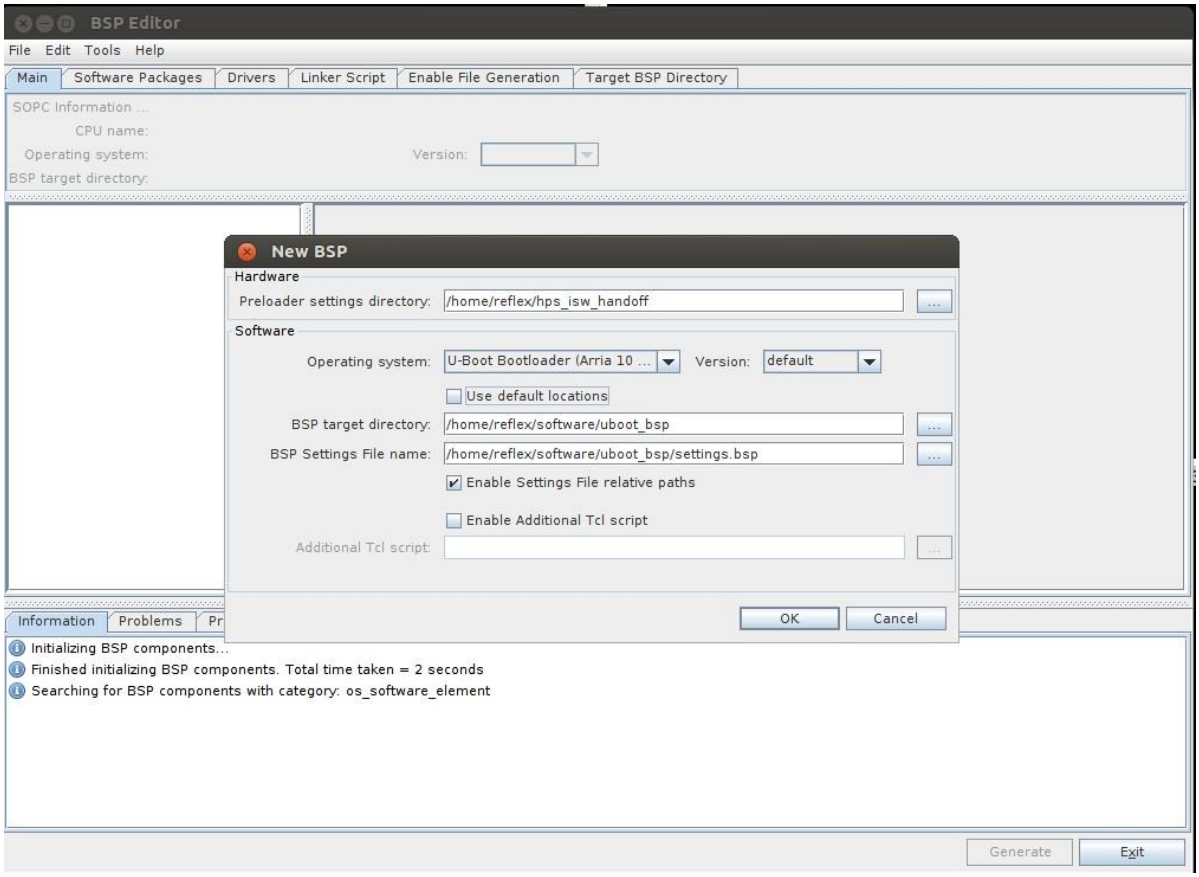
```
export UBOOT_SRC_DIR=`pwd`/uboot-src
```

4) Launch the bootloader generator with the command:

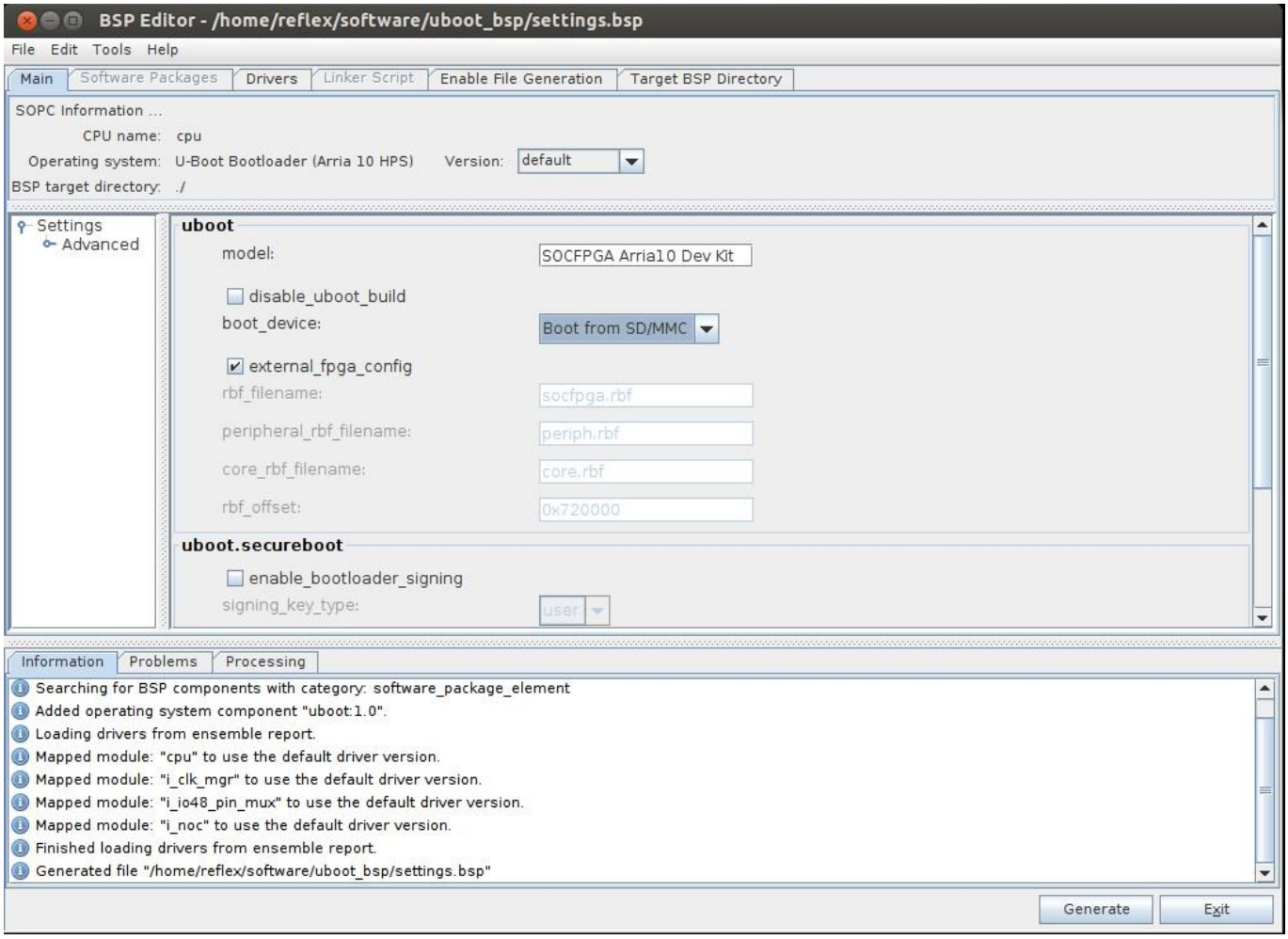
```
./bsp-editor.sh
```

   a Click **File | New HPS BSP** to display the New BSP window.
   b In the hardware section, click the browse button to select the handoff directory.
   c In the software section, you can leave the **Use default location** checkbox checked to create a directory called uboot at the handoff directory level in which the files generated by bsp-editor will be stored.
   You can also select your own directory in the *BSP target directory* field.

   *NOTE: Do not modify the BSP settings file name.*




d Click **OK** to validate your selection and click **OK** again to confirm that you want to overwrite the BSP directory. The following window is then displayed:




e In this window, click the boot device list box and select **boot from SD/MMC**

f Then check the **external_fpga_config** checkbox.

g Click the **Generate** button.

h Then click the **Exit** button at the bottom of the window

You should now see four generated files in the BSP target directory:

- *config.mk*: a configuration file
- *devicetree.dts*: the U-Boot device tree source file
- *Makefile*: a Makefile to build the U-Boot
- *uboot.ds*: a script for ARM DS-5 that enables you to copy the U-Boot directly to the on-chip RAM using JTAG. This file is very useful during U-Boot development.

5) Replace the BSP target directory *Makefile* with the USB *Makefile* located in <USB_Key>/HPS/U-Boot/src.

6) In the terminal, from the BSP target directory, execute the command *make* to build the U-Boot.

This creates a link in uboot-socfpga that points to the uboot-src directory, then compiles the source code to create the *uboot_w_dtb-mkpimage.bin* file.

*NOTE:*

- *The toolchain used to cross-compile the U-Boot is the one pointed by CROSS_COMPILE variable.*
- *To clean the build, execute the make clean command.*
- *To clean only the source code tree, execute the make clean-uboot command.*

### 8.7.5 Kernel Unitary Build

All this chapter is available on **Linux host system only**.

As shown in the global build flow, the Linux kernel source code is retrieved from a git repository. Unlike the U-boot repository, this repository is now archived, but you can download the tarball at this address: https://github.com/altera-opensource/linux-socfpga/archive/9689ce4f759ad0f13f6911d1f1309384f2b05f91.tar.gz. It corresponds to the tag 'rel_socfpga-4.1.22-ltsi_16.10.02_pr' of the *socfpga-4.1.22-ltsi* branch that is the commit 9689ce4f759ad0f13f6911d1f1309384f2b05f91. A kernel patch is applied to this kernel to adapt it to the Achilles board.

#### 8.7.5.1 Source Code Modifications

The kernel.patch performs the following modifications:

- Creates the new kernel configuration file socfpga_rxc_defconfig
- Adds devicetree support to the temperature sensor TMP102AIDRLR driver
- Edits AT24 EEPROM driver probe function and device tree parsing.

#### 8.7.5.2 Kernel Image

To build the zImage file:

0) Be sure that the SOCEDS_ROOT_PATH, CROSS_COMPILE and ARCH variables are already set in your shell environment as explained in previous chapter 8.7.1.

1) Download kernel source files from GitHub archive, and apply the patch using the following script:

```
./linux_download.sh
```

2) Execute the commands:

```
cd linux-src
make socfpga_rxc_defconfig
make zImage
```

The zImage is then created in the arch/arm/boot directory.

*NOTE: You can run the make mrproper command if you want to delete the current configuration and all generated files.*

#### 8.7.5.3 Kernel Device Tree

The device tree source file *linuxDT.dts* provided with the board package has been manually customized for Achilles.

You can use this .dts file to generate the *linuxDT.dtb* binary file:

1) Then execute the command:

```
dtc -I dts -O dtb -o linuxDT.dtb linuxDT.dts
```

*NOTE: If you prefer to create and customize your own linuxDT.dts file, you can generate the file by executing the command:*

```
sopc2dts --input hps.sopcinfo --output linuxDT.dts --board hps_a10_common_board_info.xml --board hps_a10_devkit_board_info.xml --board ghrd_10as066h4f34e3sg_board_info.xml --bridge-removal all --clocks
```

## 8.8 Board Power Up

### 8.8.1 BSEL Configuration

When the board is powered on, the BSEL pin must be set to 0x05, as shown below, so that the BootROM can load the U-Boot from eMMC. If the BSEL pin is not set to 0x05, the board cannot boot the HPS.

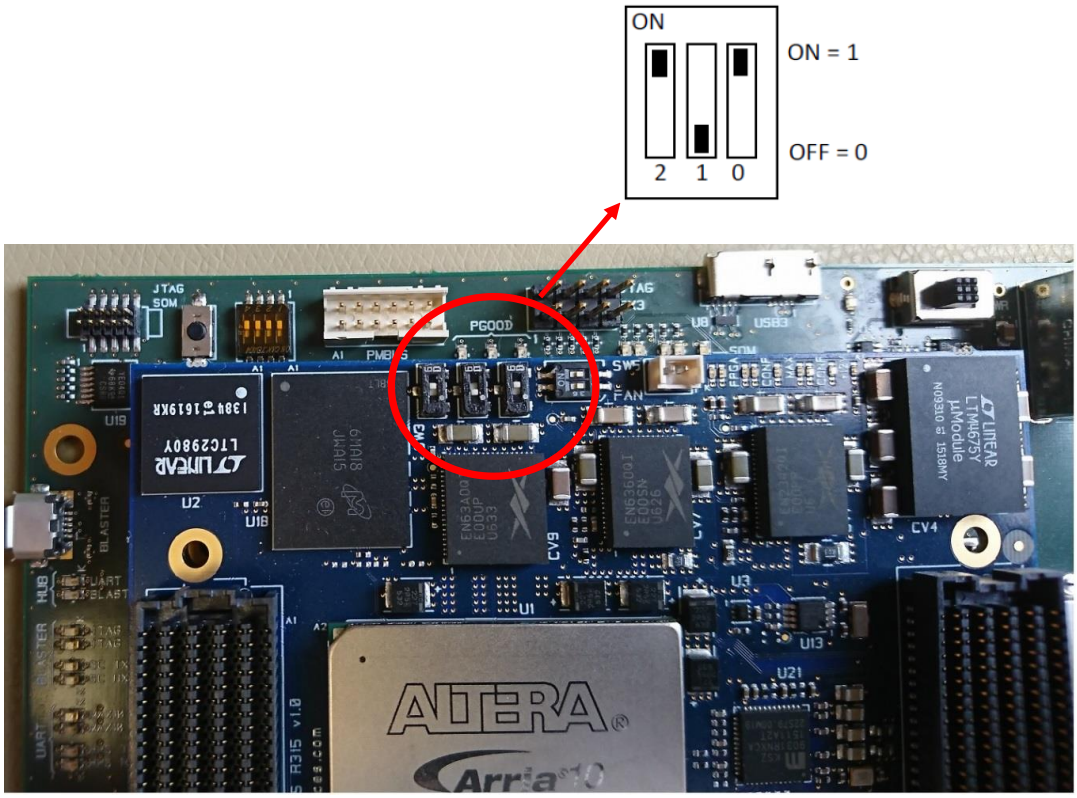
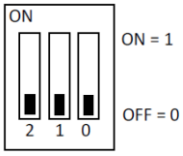
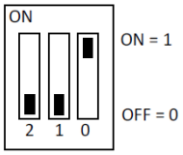
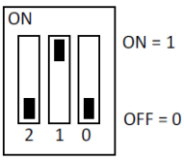
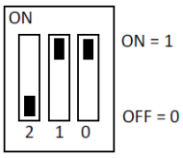
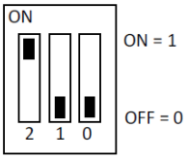
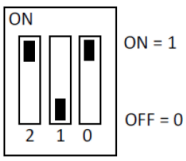
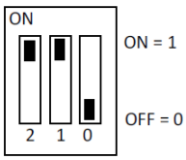
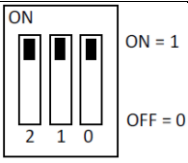



*Figure 44: BSEL location*

The full BSEL value device mapping is shown below:

| BSEL (HEX) | SWITCH POSITION | STATE POSITION | DEVICE |
|---|---|---|---|
| 0x00 |  | BSEL 2 = "OFF"<br>BSEL 1 = "OFF"<br>BSEL 0 = "OFF" | Reserved |
| 0x01 |  | BSEL 2 = "OFF"<br>BSEL 1 = "OFF"<br>BSEL 0 = "ON" | Boot from FPGA (HPS-to-FPGA bridge) |
| 0x02 |  | BSEL 2 = "OFF"<br>BSEL 1 = "ON"<br>BSEL 0 = "OFF | 1.8V NAND flash memory |
| 0x03 |  | BSEL 2 = "OFF"<br>BSEL 1 = "ON"<br>BSEL 0 = "ON" | 3.0V NAND flash memory |
| 0x04 |  | BSEL 2 = "ON"<br>BSEL 1 = "OFF"<br>BSEL 0 = "OFF" | 1.8V SD/MMC flash memory |
| 0x05 |  | BSEL 2 = "ON"<br>BSEL 1 = "OFF"<br>BSEL 0 = "ON" | 3.0V SD/MMC flash memory |
| 0x06 |  | BSEL 2 = "ON"<br>BSEL 1 = "ON"<br>BSEL 0 = "OFF" | 1.8V QSPI flash memory |
| 0x07 |  | BSEL 2 = "ON"<br>BSEL 1 = "ON"<br>BSEL 0 = "ON" | 3.0V QSPI flash memory |




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

### 8.8.2 Serial Connection

You can connect a PC to the HPS using the USB connector. This enables the retrieval of serial log messages from the HPS, with an FTDI component on the board performing RS232 / USB conversion.

To make this serial connection work, an FTDI driver must be installed on the PC. This driver can be installed during the installation of the Altera SoC Embedded Design suite or downloaded from the Internet.

When the board is powered-on:

- If you are using **Windows**, you must identify which COM port is used for the HPS. To do this, open the device manager; you will see two COM ports; one for the MAX® 10 connection and the other for the HPS connection. To check which COM port is used for HPS, use a serial connection client tool like *putty*, and establish a connection to a port using the following parameters:
  - Speed: 115200
  - Data bits: 8
  - Stop bits: 1
  - Parity: none
  - Flow control: none

  Then press any key on the putty terminal and observe your Achilles starter board. There are four UART LEDs beside the USB connector: HPS RX, HPS TX, MAX10RX, MAX10TX. When you press the key in the putty terminal, the HPS TX LED will blink if you are connected to the HPS COM port. If the MAX10TX LED blinks instead, then use the other port to communicate with the HPS.

- If you are using **Linux**, use the *dmseg* command to detect peripheral devices. You should see messages in the command output like: "FTDI USB Serial Device converter now attached to ttyUSBX" where X can be any number. Use the same method as described above for Windows to check which COM port corresponds to the HPS device.

If the connection is established quickly after power up, you should see U-Boot and Kernel boot log messages. After all these boot messages, a Linux prompt should be displayed in the terminal.

### 8.8.3 Ethernet Connection

You can also connect your PC to the HPS using an Ethernet connection though SSH, and an SSH client tool like *putty* (for both Windows and Linux). Unlike serial connection, however, Ethernet connection requires a delay after power-up and before trying to connect to leave enough time for the Linux boot to complete.

The default Linux network setup is:

- IP address: 192.168.1.155
- Netmask: 255.255.255.0
- Gateway: 192.168.1.1

On the Busybox root file systems used during factory boot flow, a root user has been created, which uses the password “root”. The kernel is configured to retrieve IP address by DHCP so you need first to connect to the board using serial connection to know the board IP address. Assuming it has taken IP address 192.168.1.42, the SSH command to connect with the root user is: *ssh root@192.168.1.42*.

On the custom Linaro root file system, there is a linaro user with the password “linaro”. The SSH command to connect with this user is: *ssh* linaro@192.168.1.155*.*
To get root access using SSH, additional configuration steps must be done: enabling the “root” user and then authorizing the root login in the SSHD configuration file. To achieve this, following procedure must be followed:

- Connect to the board using serial connection or SSH logged as Linaro user.
- Execute following command to enable “root” user: “sudo passwd root” then enter and confirm your new root password
- Authorize root access for SSH by editing SSHD configuration file: open the file with root right executing the command “sudo vi /etc/ssh/sshd_config” then add the line “PermitRootLogin Yes”.
- Reboot the board
- You can now connect to the board with root access using: *ssh root@192.168.1.155.*

*NOTE: The SSH command will only work if your PC IP address is 192.168.1.XXX, where XXX is a value between 1 and 255 that is not 155.*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 8.9 HPS Test

You can test that all peripherals supported by the HPS are working either by using the Linux test scripts and programs that have been manually added to the Linaro-nano root file system or by using the overlay folder for the Busybox-based root file system.

The following table describes each of these tests:

| Test | Description |
|---|---|
| **Benchmark** | Two benchmark applications are used to test CPU and memory performance: Coremark and STREAM.<br>• You can launch the Coremark benchmark by executing the */root/coremark.sh* script. This script creates a log file containing the test results. Check the Iterations/Sec value in this log file.<br>• You can launch the STREAM benchmark by executing the */root/stream* binary. The test result is directly output in the terminal. |
| **EEPROM** | You can test EEPROM using the script /root/*eepromtest.sh*. This script writes a pattern into the EEPROM start address using the user interface via the /sys file system, then reads it back to check that it was successfully written. |
| **HPS-to-FPGA bridges** | HPS-to-FPGA bridges are pure SoC implementation and are therefore out of the scope of REFLEX CES. |
| **LEDs** | You can test the two LEDs connected to dedicated GPIOs using the script */root/ledtest.sh*. This script turns on and off LEDs using the user interface via the /sys file system. |
| **Memory** | You can test DDR4 when Linux boots for the first time and by using the script */root/memtest.sh*. This script uses the memtester utility to test available memory. The script can only test about 2GB due to the limitation of available virtual address space per process, however. It starts from address 0, but you can edit the script to add an offset using the -p parameter. |
| **RTC** | You can test the RTC using the script */root/rtctest.sh*. This script prompts you to enter a date using the date command, then writes this date to the RTC. It reads back the RTC to display the date and prompts you to validate it. |
| **Temperature Sensor** | The temperature sensor is tested using */root/tempsensortest.sh*. This script simply executes the sensor command to display the temperature. |
| **USB OTG** | The USB OTG test script */root/usbtest.sh*:<br>1) Prompts you to insert a USB key.<br>2) Creates a new file and writes a fix pattern.<br>3) Reads the file contents back to check for errors.<br>4) Deletes the file and removes the USB key. |

## 8.10 Development Process

### 8.10.1 Loading U-Boot in OCRAM

During the development stage, before you can boot a full system to create eMMC partitions and store binaries in the eMMC (factory build flow), you must first load the U-Boot in OCRAM.
Although it is possible to create a U-Boot hex file and a HDL design to boot from the FPGA as for the factory boot flow, it is not an optimum solution at this stage as it takes a lot of time to modify and test the U-Boot source code.
Fortunately, you can load the U-Boot binary directly in OCRAM using ARM DS-5 and the JTAG probe, as BootROM does when the HPS goes out of reset.

To do this:

1) When the U-Boot is built, as well as the *uboot_w_dtb-mkpimage.bin* that is generated, the following two files are also created:
   - *<BSP_target_directory>/devicetree.dtb*: the U-Boot device tree
   - *<BSP_target_directory>/uboot-socfpga/u-boot*: the U-Boot image at the root path of U-boot source code.

   Copy these two files and the <BSP_target_directory>/uboot.ds file to a directory on your PC.

2) Connect the JTAG probe to the PC and the Achilles DevKit (the USB Blaster in this case).

3) Open ARM DS-5 and create a debug configuration:
   a Click the **Run** | **Debug Configurations**… menu.
   b On the left panel, click the DS-5 Debugger menu, then the icon on the left at the top of this panel to create a new configuration. Name this configuration.
   c In the **Connection** tab:
      - In the **Select Target** section, select the Debug Cortex-A9x2 SMP as the target in the Linux Kernel and/or Device Driver Debug.
      - In **Target Connection**, select your probe.
      - In the **Connections** section, click the browse button and select your detected probe.
   d In the **Debugger** Tab:
      - In the **Run Control** section, click the "connect only" radio button. Then click the "Run target initialization debug script (.ds / .py)" checkbox and select your uboot.ds file.
      - In the **Paths** section, for the source search directory configuration, point to the root path of your U-Boot source code.

4) Click the **Debug** button to validate your configuration. In the main display, you should see your debug configuration on the left under the Debug Control View. If you can't see it, select **Debug Control** from the list in **Window** | **Show Views** menu.

5) Before powering on the board, change the BSEL pins so that the BootROM does not load U-Boot from the eMMC. Set the BSEL pins to BSEL[2:0] = 0x2 (BSEL0 and BSEL2 in the down position and BSEL1 in the up position). This sets the boot from the NAND flash, which is not possible with the Achilles DevKit.

6) Power on the board.

7) An optional but recommended step is to integrate a terminal in the tool to receive serial output. Click the **Windows | Show view | Others** menu, then select the terminal view. In the terminal view, click the Setting icon to configure your serial connection. Then click the Connect icon.

8) In the Debug Control view, click the icon on the left to connect to the target. This executes the uboot.ds script, which loads the U-Boot in OCRAM.

*NOTE: Just after the load, the Cortex-A9 processor may stop on address 0x00000000. If this happens, click the icon Disconnect from target, and then click the Connect icon again.*

9) As well as loading the U-Boot in OCRAM, the uboot.ds script enables you to set breakpoints in the U-Boot source code. At the end of the script, you can add a breakpoint in the same way as you would do with gdb on Linux; *b function_name* or *b source_code_line,* so that after loading U-BOOT, ARM DS-5 will break the processor and the source code will be displayed:

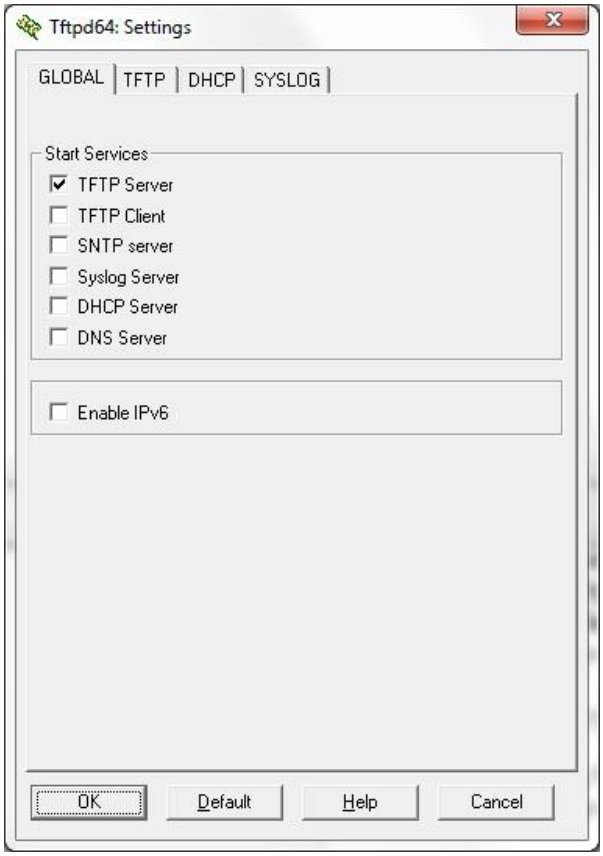
### 8.10.2 Loading Kernel through TFTP

During kernel development, for the same reasons as described in the previous section for the U-Boot, it is not advisable to boot a full system to copy zImage into dedicated partitions.
Consequently, the best solution is to boot the kernel through TFTP in U-Boot. To do this, the kernel image must be copied in a specific directory managed by a TFTP Server. It is then sent over the network to the board and directly copied into DDR.

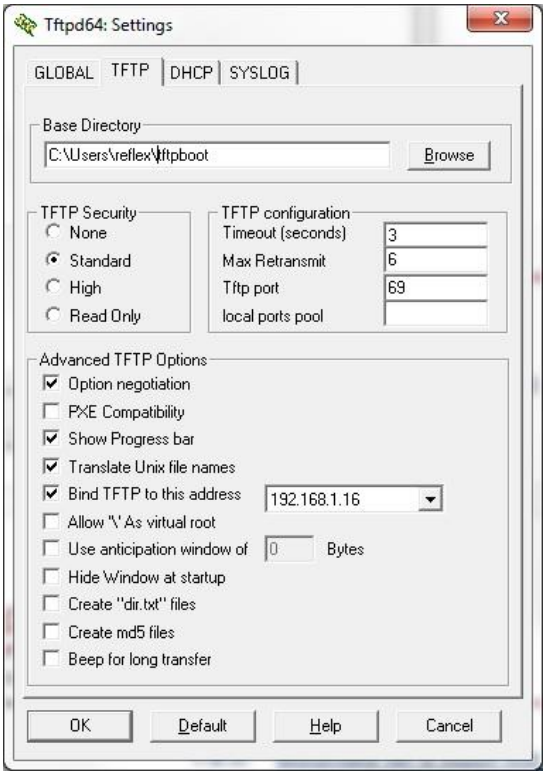
To configure this:

1) Install a TFTP server on your PC.

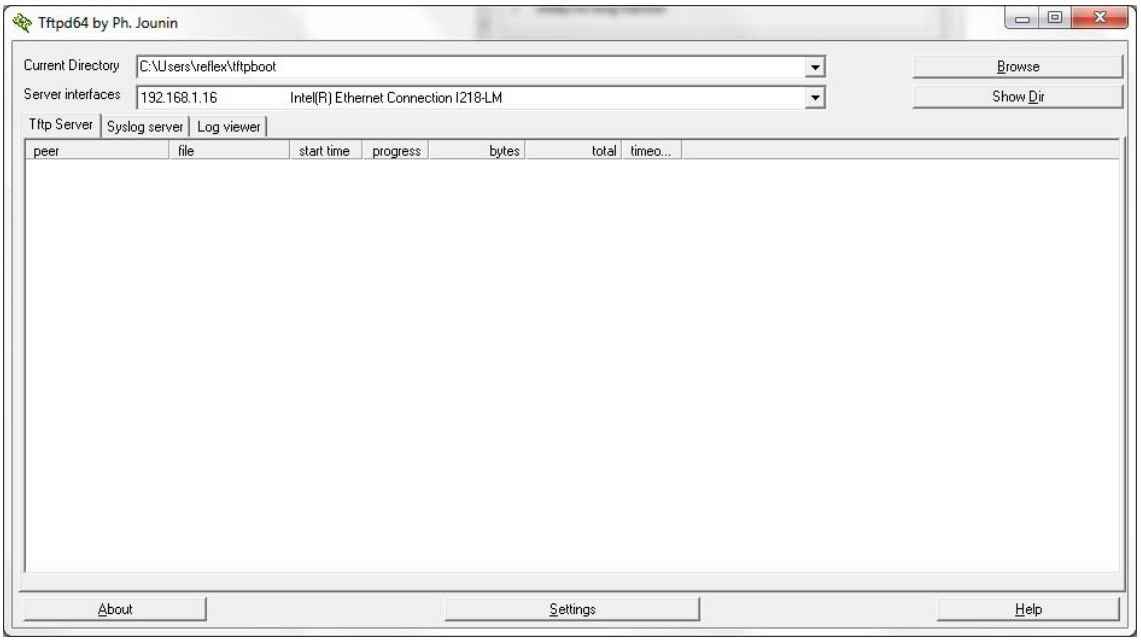
- For **Windows** OS, we recommend a simple and easy-to-use software tool called Tftpd64. This contains a TFTP, SNTP, DHCP, Syslog and DNS server, but as the other server options are not required, you can select only the TFTP server in the Settings dialog box, as shown below:



In the TFTP settings menu, set your base directory where the kernel image will be stored. You can also specify the address the TFTP server will bind to in the Advanced TFTP Options section:




In the main window, select the server interface:



- For **Linux** OS, you can install either tftpd or atftpd. Documentation and configuration files for these tools can be found on the Internet.

2) Copy your kernel into your TFTP server.

3) You must next change the U-Boot boot command to retrieve the kernel image through TFTP instead of the image stored in the eMMC partition. To do this, edit the configuration file *socfpga_arria10.h*.

   First, change the variable *serverip* to match the server PC IP address.

   The boot command is managed by the definition CONFIG_BOOTCOMMAND, which uses two variables of the CONFIG_EXTRA_ENV_SETTINGS definition. One defines how the kernel is retrieved and the other how the kernel mounts the root file system. Several configurations are possible and these are managed by the following REFLEX CES configurations:
   - *RXC_CONFIG_BOOT_FROM_MMC* involves the use of the *rxcmmcload* and *rxcmmcboot* variables. The first enables the kernel to be loaded from the second eMMC partition; the second enables the root file system stored in the third eMMC partition to be mounted. This is the default configuration.
   - *RXC_CONFIG_BOOT_FROM_NET* involves the use of the *rxcethload* and *rxcethboot* variables. The first enables the kernel to be loaded over TFTP, the second enables the NFS root file system stored in the third eMMC partition to be mounted.
   - *RXC_CONFIG_BOOT_HYBRID* involves the use of the *rxcethload* and *rxcmmcboot* variables. The first enables the kernel to be loaded over TFTP; the second enables the root file system stored in the third eMMC partition to be mounted.

   To retrieve the kernel image through TFTP, therefore, use the *rxcethload* variable. This variable uses the dhcp U-Boot command so that the board can obtain an IP address by DHCP. If you also plan to use the NFS root file system (see Section 8.10.3, below) then you can just use the *rxcethboot* variable in your boot command (that is, the *RXC_CONFIG_BOOT_FROM_NET* configuration described above).

4) Then rebuild the U-Boot image and copy it into the first eMMC partition or load it in OCRAM.

### 8.10.3 Mounted NFS Root File System

It is more convenient to mount the root file system over the network during tests or when updating eMMC partitions. When updating eMMC partitions, in particular the one dedicated to the root file system, it is better not to mount the root file system at all. In such cases, using an NFS server can be very helpful.

To mount an NFS root file system, you must install an NFS server on your PC (Linux only) and NFS support must be enabled and configured in U-Boot:

1) On your PC, install an NFS server. If your PC is running Ubuntu, you can install the nfs-kernel-server packet.

2) Add your root file system directory to the export configuration files: edit the file */etc/exports* to add the following line:
*/tftpboot/linaro-rootfs *(rw,no_root_squash)* replacing '/tftpboot/linaro-rootfs' with your own directory path.

3) Restart the NFS server to take the new configuration into account using the command: *sudo service nfs-kernel-server restart*

4) As explained in Section 8.10.2, above, in the *socfpga_arria10.h* file, the *rxcethboot* variable enables you to mount the NFS root file system defining the kernel boot command. Check that the following parameters are set in the bootargs; if not, edit these parameters as follows:
   - root=/dev/nfs: indicates you are using NFS
   - rw: the root file system is mounted with read-write access.
   - nfsroot=192.168.1.34:/tftpboot/linaro-rootfs,nolock,rsize=4096,wsize=4096 : indicates the NFS IP server address and the path on the server to the exported directory (the same path that has been added to the file /etc/exports).

5) Edit the CONFIG_BOOTCOMAND to obtain the configuration you want for kernel loading and root file system mounting, as described in Section 8.10.2, above.

6) Then rebuild the U-Boot image and copy it into the first eMMC partition or load it in OCRAM.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

## 8.11 How to Restore / Update eMMC Partition Content

This section summarizes the steps required to restore the eMMC with default content or to update partitions. If you need more information about a particular step, refer to the relevant sections in this manual.

### 8.11.1 Restoring eMMC Partitions

To restore the eMMC partitions, you must enforce a factory boot flow. You can do this using Autobuild:

1) Launch the command "./*build.sh all*", which generates all the software binaries that will be stored in the eMMC in the folder *Autobuild/buildroot-2015.08/output/images*: U-Boot, the Linux kernel and device tree. It also copies the Linaro-based root file system to this folder.

2) Then copy the generated binaries to your TFTP server:
   - Rename *uboot_w_dtb_mkpimage.bin* to *uboot.bin*
   - Rename Linux kernel *zImage* to *zImage.default*
   - Rename the Linux device tree *linuxDT.dtb*
   - Rename the root file system to *rootfs.tar.gz*

3) Launch the command "./*build.sh clean"* to clean the previous build.

4) Now launch the command ./*build.sh factory manual*, which generates the U-Boot in hex format and the Linux kernel, including the RAM-based root file system that will automatically create the eMMC, in the folder *Autobuild/buildroot-2015.08/output/images*: If you want to restore the eMMC to its pure factory state, then you do not need the .hex file as the HDL design "RefDesign_HPS_boot_from_FPGA.sof" is already provided in the DevKit package. Otherwise you will have to generate this design again using the new .hex file.

5) Copy the generated zImage to your TFTP server.

6) Power up your board and program the FGPA with the RefDesign_HPS_boot_from_FPGA.sof file. It can be helpful to use a serial connection to see HPS traces during this procedure. When the FPGA is programmed, the HPS reboots automatically and the factory boot flow starts:

   - U-boot is loaded in the OCRAM by the HDL design, which then now retrieves *zImage* from the TFTP server.

- zImage loads its RAM root file system
- Edit the partitioning script */root/partionning.sh* to make it works with your TFTP server settings (vi editor is present on the RAM file system). The provided script works with following configuration:
  - TFTP server IP address: 192.168.1.200
  - All binaries are stored in /tftpboot/SomA10-V2/<boardName> where boarName is the value contained in /sys/firmware/devicetree/base/rxcboard of RAM file system.
- Executes *./root/partionning.sh.*
  The script deletes all existing partitions and creates new partitions. It then retrieves zImage and linuxDT.dtb from the TFTP server to write them into the second eMMC partition. It retrieves rootfs.tar.gz by TFTP and then extracts it to the third partition. Finally, it retrieves uboot.bin to create the first partition using a "DD" command.

When the eMMC partitions have been restored, you can power-down and power-up your board. The HPS then boots in default boot flow mode using the default binaries from the different eMMC partitions.

*NOTE: Here the build.sh script has been called with "factory manual" parameters not to automatically execute the partitioning script. You can also choose to first edit the partitioning script located in Autobuild/buildroot-inputs/rootfs-overlay-factory/root/. to match your TFTP server settings, then execute "./build.sh factory auto" to generate a zImage file that will automatically execute the partitioning script at boot.*

### 8.11.2 Updating an eMMC partition

The best way to manually update an eMMC partition is to change the default boot flow.

Instead of using the kernel and device tree from the second eMMC partition, and the default root file system from the third partition, you can load the kernel and device tree through TFTP and mount the NFS root file system as described in Section 8.10.

You can then connect to the HPS using a serial or Ethernet connection and manually update the eMMC partition. For example, to update LinuxDT.dtb or zImage in the second eMMC partition:

1) On your host system, copy the file you want to update to your TFTP server or to your NFS root file system.

2) From the terminal on the target, mount the eMMC partition and move to this folder using the command: *mkdir /mnt/temp ; mount -t vfat /dev/mmcblk0p2 /mnt/temp ; cd /mnt/temp*

3) If the file to update is on the TFTP server, download the file with the command: *tftp -g -r file <your_tftp_server_ip_address>*. You can also push the file to the target from the TFTP server with the SSH command: *scp file root@192.168.1.155:/mnt/temp*.

If the file has directly been copied to the NFS file system on the host PC, you have direct access to the file on the target, so copy it from its source path.

4) Execute the *reboot* command. After the reboot and default boot flow, the new kernel image or device tree is loaded.

If you want to change the whole root file system, the same procedure is used, except the mount command *mount -t ext3 /dev/mmcblk0p3 /mnt/temp* is used instead as it is not the same partition. You can then extract the root file system archive to */mnt/temp*.

To update U-boot, you do not have to mount the partition, but can just copy the file into your target and execute the following command: *dd if=uboot.bin of=/dev/mmcblk0p1 bs=512 count=2048.*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

# 9. Component references

The table below provides the references for the components used on the SoM board:
Components may change to overcome obsolescence.

| Component | Manufacturer | Reference |
|---|---|---|
| **Clock** | | |
| Low Jitter Clock Generator | SiLab | Si5341A-D |
| **DDR4 FPGA** | | |
| DDR4 (SoM Turbo) | Micron | MT40A1G16WBU-083E:B |
| DDR4 (SoM Indus) | Micron | MT40A512M16JY-083E IT:B |
| DDR4 (SoM Light) | Micron | MT40A512M16JY-075E:B |
| **USB 3.0** | | |
| FX3 SuperSpeed controller | Cypress | CYUSB3013-BZXC |
| **USB 2.0** | | |
| USB 2.0 ULPI Transceiver | Microchip | USB3320C-EZK |
| **GbE** | | |
| GbE Eth Transceiver (RGMII) | Microchip | KSZ9031RNXCA |
| **HPS** | | |
| DDR4 (SoM Turbo) | Micron | MT40A1G16WBU-083E:B |
| DDR4 (SoM Indus) | Micron | MT40A512M16JY-083E IT:B |
| DDR4 (SoM Light) | Micron | MT40A512M16JY-075E:B |
| EMMC | Micron | MTFC32GAKAENA-4M IT |
| EEPROM | STMicroelectronics | M24256-BFMC6TG |
| RTC | NXP | PCF8563TS/5 |
| Temperature Sensor | Texas Instruments | TMP102AIDRLR |
| **LPC+ / HPC Connector** | | |
| HPC Connector TOP | Samtec | ASP-134486-01 |
| HPC Connector BOP | Samtec | ASP-134602-01 |

| Component | Manufacturer | Reference |
|---|---|---|
| LPC+ Connector TOP | Samtec | ASP-134486-01 |
| LPC+ Connector BOT | Samtec | ASP-134602-01 |

*Table 22: Example of SoM Board Implementation References*




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com

# 10. TROUBLESHOOTING

For troubleshooting questions, please contact REFLEX support at https://support.reflexces.com/.




REFLEX CES Europe, Japan
Email : sales@reflexces.com
REFLEX CES Americas / Asia Pacific
Email : salesusa@reflexces.com