SCR1 Software Development Kit. User Guide

v1.0, 2017-12-14

Table of Contents

Revision History

Version	Date	Description
1.0	2017-08-01	Initial revision

This is a user guide for the Arria V GX based SCR1 SDK Arria V GX FPGA Starter Kit from the Altera.

1. Setup equipment

Arria-V based SCR1 SDK HW platform consist of three mandatory components:

1. Arria V GX FPGA Starter Kit
   https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-arria-v-starter.html
2. CP2104-MINIEK
   http://www.silabs.com/products/development-tools/interface/cp2104-mini-evaluation-kit
3. ARM-USB-OCD-H
   https://www.olimex.com/Products/ARM/JTAG/ARM-USB-OCD-H/

Additionally, you may need some wires and cables:

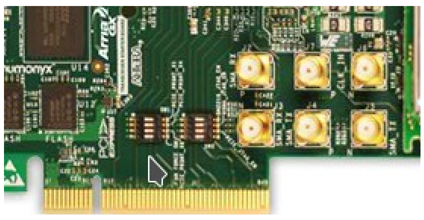
1. Wires: 3 mounting wires
2. Cables:
   - USB A (m) - mini USB B (m)
   - Ethernet cat.5E

1.1. Prerequisites

FPGA board image update is done over the network, so you'll need access to the local network with DHCP capability. Besides of the Arria-V board, you'll need PC, connected to the same network.

IMPORTANT

1. Before the upgrade, please, make sure the board cooling Fan is ON. If not, turn on Fan before power on.
2. To switch on board cooling Fan, move FAN_FORCE DIP switch to ON position at SW. Corresponding DIP is marked by cursor in the figure below



2. SDK HW assembly

2.1. Connecting serial console

In order to get access to the board console, it is required to mount *HSMC Debug Header Breakout Board* (included in Arria-V Starter Devkit) and connect *CP2104-MINIEK* USB-to-UART converter to the breakout board with external wiring, as described in this section.

2.2. Pins assignment

- Connect CP2104 pin GND to the GND pin on the HSMC Debug board
- Connect CP2104 pin TXD to the 3 pin on the HSMC Debug board
- Connect CP2104 pin RXD to the 4 pin on the HSMC Debug board

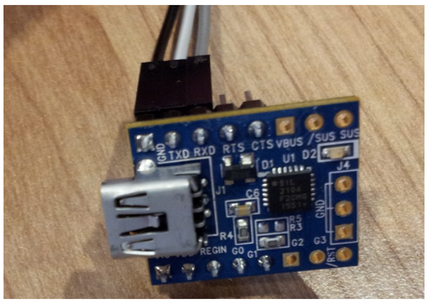
- Connect JTAG pin GND to the 38 pin on the HSMC Debug board
- Connect JTAG pin GND to the 36 pin on the HSMC Debug board
- Connect JTAG pin GND to the 34 pin on the HSMC Debug board
- Connect JTAG pin GND to the 32 pin on the HSMC Debug board
- Connect JTAG pin GND to the 30 pin on the HSMC Debug board
- Connect JTAG pin GND to the 28 pin on the HSMC Debug board
- Connect JTAG pin GND to the 26 pin on the HSMC Debug board
- Connect JTAG pin GND to the 24 pin on the HSMC Debug board
- Connect JTAG pin VREF to the 22 pin on the HSMC Debug board
- Connect JTAG pin VREF* to the 21 pin on the HSMC Debug board
- Connect JTAG pin SRSTn to the 35 pin on the HSMC Debug board
- Connect JTAG pin TDO to the 33 pin on the HSMC Debug board
- Connect JTAG pin TCK to the 29 pin on the HSMC Debug board
- Connect JTAG pin TMS to the 27 pin on the HSMC Debug board
- Connect JTAG pin TDI to the 25 pin on the HSMC Debug board
- Connect JTAG pin TRSTn to the 23 pin on the HSMC Debug board

IMPORTANT

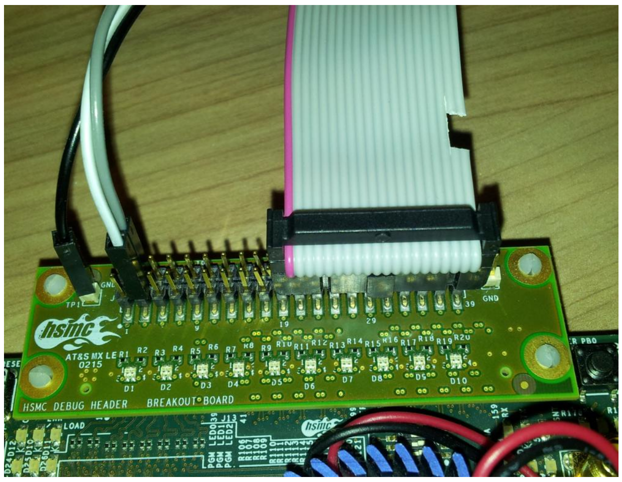
You can either connect JTAG port using mounting wires, or apply the existing flat cable from the Olimex kit. For the Olimex kit, some rework will be needed, it also affects couple of unused pins on the header. It is mandatory to cut two wires (19 and 20) in the Olimex cable, as shown below.

The following figures assume the re-worked Olimex cable is used:

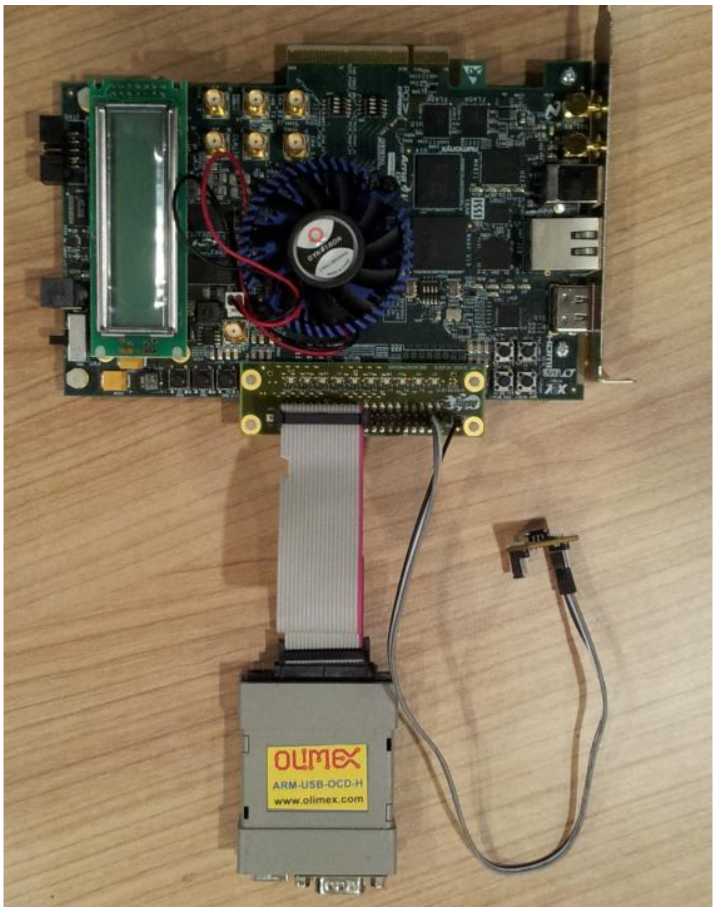
1. Wires connection to the CP2104



2. HSMC Debug Board mounting to the Arria-V HMSC header

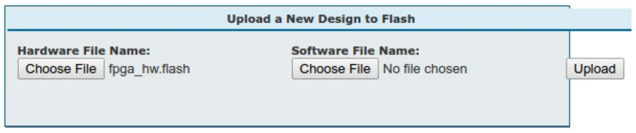
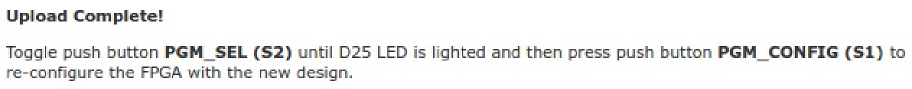


3. Resulting setup



3. Arria-V HW image update

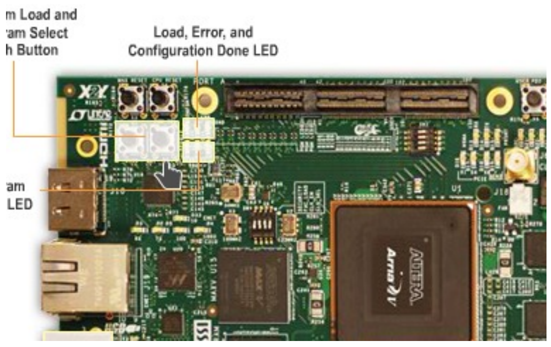
1. Connect board by the ethernet-cable to the network with DHCP-server
2. Power on the board and wait for IP-address to appear at the board LCD-display (IP address may be different):

   192.168.1.52
3. Open web browser at PC and open http-connection the IP-address from LCD
4. You will see Board Update Portal page.
5. Press button “Choose File” and set proper file for "Hardware File Name:” (as shown on the Figure), then press the "Upload" button
   
   Upload a New Design to Flash
   Hardware File Name: Choose File fpga_hw.flash
   Software File Name: Choose File No file chosen
   Upload
6. Wait for upload completion:
   
   Upload Complete!
   Toggle push button **PGM_SEL (S2)** until D25 LED is lighted and then press push button **PGM_CONFIG (S1)** to re-configure the FPGA with the new design.
7. Power off the board. Flash update is complete.

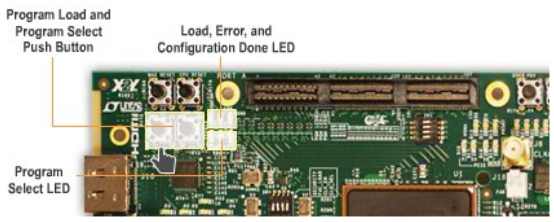
4. Booting the new FPGA image

IMPORTANT	**After FGPA image updated successfully, in order to load the new image, you have to do the following sequence after every power on!** *Otherwise (by default), system will boot into standard board update portal image.*

1. Press **PGM_SEL (S2)** one time so D25 LED is lighted green.
   That selects new image to boot into - otherwise you'll load the default image with Altera board update portal SW.
   Corresponding button is marked by cursor in the figure below:



2. Press **PGM_CONFIG (S1)** once to load the selected image and check **CONFDN (D12)** led is active
   Corresponding button is marked by cursor in the figure below:

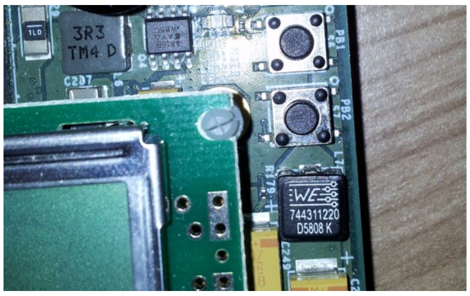


3. If **CONFDN(D12)** led is not active green and/or ERR (D10) led is active red after this step, repeat the flash update sequence, as described in the section "Arria-V HW image update"

5. Resetting the board:

NOTE

Press **PB2** button if you need to reset board to the bootloader.
Corresponding button is in the figure below:



6. UART connection settings

NOTE

- Bps/Par/Bits - 115200 8N1
- speed - 115200
- bits - 8
- stop bits - 1
- parity - none
- Hardware Flow Control: No

7. Using UART terminal

1. Connect PC to the uart port and open any terminal (minicom is used in the example below) After reset or FPGA firmware update you will see the bootloader prompt:

```
SCR loader v1.0-scr1_RC
Copyright (C) 2015-2017 Syntacore. All rights reserved.
ISA: RV32IMC [40001104] IMPID: 17101000
BLDID: 17121400
Platform: a5_scr1, cpuclk 30MHz, sysclk 30MHz
Memory map:
00000000-0FFFFFFF       00000000        DDR
F0000000-F001FFFF       00000000        TCM
F0040000-F0040FFF       00000000        MTimer
FF000000-FF0FFFFF       00000000        MMIO
FFFF0000-FFFFFFFF       00000000        On-Chip RAM

1: xmodem load @addr
g: start @addr
d: dump mem
m: modify mem
i: platform info
:
```

1. If you press **"i"** button you can see additional info about the platform

```
ISA: RV32IMC [40001104] IMPID: 17101000
BLDID: 17121400
Platform: a5_scr1, cpuclk 30MHz, sysclk 30MHz
Memory map:
00000000-0FFFFFFF       00000000        DDR
F0000000-F001FFFF       00000000        TCM
F0040000-F0040FFF       00000000        MTimer
FF000000-FF0FFFFF       00000000        MMIO
FFFF0000-FFFFFFFF       00000000        On-Chip RAM
Platform configuration:
FF010000        irq 0   UART16550
:
```

7.1. Load binary images to the Memory address

TIP *SCR bootloader supports only .binary files loading using x-modem*

1. Wait for the booloader prompt

```
1: xmodem load @addr
g: start @addr
d: dump mem
m: modify mem
i: platform info
:
```

2. Press button **“1”**
3. Print required TCM address (in hex) and press **“Enter”**. **“C”** character starts to print continuously

```
xload @addr
addr: f0000000
CCCCCCCCCCCCCC
```

1. Open xmodem upload menu (for minicom terminal you need to press **“Ctrl+A”** and press **“S”**). Then select “xmodem”:

```
 +-[Upload]--+
 | zmodem    |
 | ymodem    |
 | xmodem    |
 | kermit    |
 | ascii     |
 +-----------+
```

1. Press **“Enter”**. Then select required bin-file for the loading (mark it and press **“space”** button for minicom).

```
+------------------------[Select a file for upload]-------------------------+
|Directory: /images/a5/scr1/                                                |
| [..]                                                                      |
| dhry21-o3lto.bin                                                          |
|                                                                           |
```

1. Press **“Enter”** button. Image transfer will start.

```
+-----------[xmodem upload - Press CTRL-C to quit]------------+
|Sending dhry21-o3lto.bin, 107 blocks: Give your local XMODEM |
|receive command now.                                         |
|Bytes Sent:  13952   BPS:5468                                |
|                                                             |
|Transfer complete                                            |
|                                                             |
| READY: press any key to continue...                         |
+-------------------------------------------------------------+
```

1. After loading completes, status information will be shown:

```
Xmodem successfully received 13952 bytes
```

7.2. Example: Dhrystone run from TCM memory

1. Load **dhry21-o3lto.bin** to the TCM base address (0xf0000000)
   And run test from **0xf0000200** address:

```
1: xmodem load @addr
g: start @addr
d: dump mem
m: modify mem
i: platform info
start @addr
addr: f000200
```

1. After run you will see test results

```
Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Compiler flags: -O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las  -flto
HZ 1000000, CPU MHz 30.000
Execution starts, 500 runs through Dhrystone



...



Time: begin= 48999682, end= 49004684, diff= 5002
Microseconds for one run through Dhrystone: 10.004
Dhrystones per Second:                      99960
```

7.3. Using OpenOCD

The OpenOCD (Open On-Chip Debugger) is open-source software that interfaces with a hardware debugger’s JTAG port. OpenOCD provides debugging and in-system programming for embedded target devices.

IMPORTANT When you boot from OpenOCD, you use images in .elf or .bin formats.

7.3.1. Installing OpenOCD

Official OpenOCD documentation is available at http://openocd.org/doc/html/index.html.

7.3.2. Starting the OpenOCD server

1. Setting environment variables:

```
$ export OOCD_ROOT=<Path to the OpenOCD installation directory>
```

2. Server start-up is entered in one line (Ubuntu):

```
$ sudo ${OOCD_ROOT}/src/openocd
-s ${OOCD_ROOT}/tcl
-f ${OOCD_ROOT}/tcl/interface/ftdi/olimex-arm-usb-ocd-h.cfg
-f ${OOCD_ROOT}/tcl/target/scr1.cfg
```

After execution to the current terminal, you will receive a message about the connection to the RISCV kernel:

```
Open On-Chip Debugger 0.10.0+dev-00993-g18ab525 (2017-09-22-14:31)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
init_targets
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
adapter speed: 2000 kHz
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain
connect_deassert_srst
Info : auto-selecting first available session transport "jtag". To override use
'transport select <transport>'.
scr.cpu
Info : clock speed 2000 kHz
Info : JTAG tap: scr1.cpu tap/device found: 0xdeb01001 (mfg: 0x000 (<invalid>), part:
0xeb03, ver: 0xd)
Info : IDCODE=0xDEBFF001 DBG_ID=0x00810100 BLD_ID=0x17100301
Info : Listening on port 3333 for gdb connections
```

3. Open the second terminal (terminal 2) and enter the command:

```
$ telnet localhost 4444
```

4. OpenOCD is up and ready to go. This terminal is an interactive console OpenOCD.
   The help command lists the available openocd commands

7.3.3. Downloading and running an image using OpenOCD

1. The load command is entered in the OpenOCD console (terminal 2):

```
halt ; load_image dhry21-o3lto.bin 0xf0000000 bin ; resume 0xf0000200
```

or

```
halt ; load_image dhry21-o3lto.elf 0x0 elf ; resume 0xf0000200
```

IMPORTANT The boot command assumes the location of the file in the current directory. For a different location, the name of the uploaded file must include a relative path.

2. After entering the command, the progress of the load is displayed

3. After the download is complete, the image will start with log output to the uart terminal:

```
Dhrystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Compiler flags: -O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las  -flto
HZ 1000000, CPU MHz 30.000
Execution starts, 500 runs through Dhrystone



...



Time: begin= 48999682, end= 49004684, diff= 5002
Microseconds for one run through Dhrystone: 10.004
Dhrystones per Second:                      99960
```

7.4. Arria V GX SDK memory map

Memory map is shown in the table below:

Table 1. Arria V GX SDK memory map

Base address	Size	Block name	Description
0xFFFF0000	64 KB	Onchip RAM	Internal memory
0xFF010000	32 B	UART	UART 16550
0x00000000	256 MB	DDR3	External DDR3 memory
0xF0000000	128 KB	TCM	Internal Tightly-Coupled Memory
0xFF000000	16 B	BLD ID	Build ID register (Read only)
0xFFFFFF80	4 B	MTVEC	MTVEC init value
0xFFFFFF00	4 B	RESET	RESET value

7.5. SCR1 core IRQ-mapping

The connection scheme for interrupt lines is given below:

Table 2. SCR1 core IRQ connection

IRQ line for the SCR1 core	IRQ init block
0	UART (UART 16550)
1-31	Not connected (constant level 0)

8. Building SDK FPGA-project for the Arria V GX SDK

8.1. General structure of the SDK project

The composition of the SDK folders is:

- doc - SDK and SCR1 user guides
- fpga
  - a5
    - scr1 - DE10-Lite FPGA project
      - ip - additional RTL IPs + bootloader image
        - uart - Opencores UART 16550 IP
  - arty
  - de10lite
- images
  - a5
    - scr1 - pre-built FPGA image
  - arty
  - de10lite
- scr1 - SCR1 repository, included as sub-module
  - src - SCR1 core RTL sources
- sw
  - fsbl - FPGA-bootloader
  - tests - some benchmark tests

Essential files: FPGA project file - a5_sdk.qpf (fpga/a5/scr1/a5_sdk.qpf)
Top module - a5_sdk (fpga/a5/scr1/a5_sdk.sv)

8.2. Additional requirements for compilation

FPGA build requires "Altera Quartus 13.0.1 Build 232" tool or earlier.

NOTE

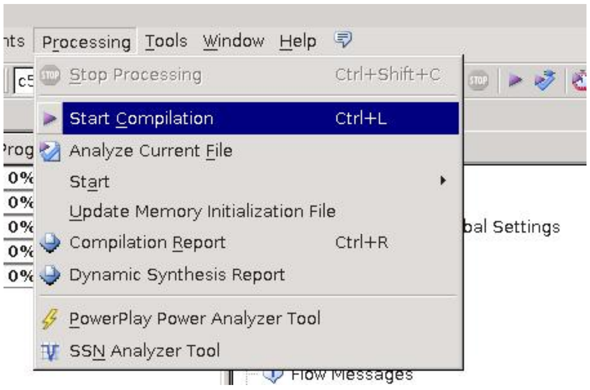
FPGA-project compilation was verified for "Altera Quartus 13.0.1 Build 232" Full version on Linux xUbuntu 16.04 with 8 GB of RAM.
Some build steps may be different for other Quartus versions.
Quartus Full version is required for a non time-limited HW firmware generation.

8.3. Building SDK FPGA project

The step-by-step FPGA project build procedure is described below:

8.3.1. FPGA firmware generation (sof-format)

- Run Quartus 13.0.1 in GUI-mode
- Select and open fpga-project file (a5_sdk.qpf)
- Press "Start Compilation" button or sellect from menu Processing → "Start Compilation"


Processing
Tools
Window
Help
Stop Processing Ctrl+Shift+C
Start Compilation Ctrl+L
Analyze Current File
Start
Update Memory Initialization File
Compilation Report Ctrl+R
Dynamic Synthesis Report
PowerPlay Power Analyzer Tool
SSN Analyzer Tool

- Wait for the compilation to complete (build time is typically 10-15 minutes)
- New "output" subfolder should appear in the "fpga" folder. It contains **a5_sdk.sof** file (Arria V GX FPGA image in sof-format).

8.3.2. Converting FPGA-image to the board flash memory image (flash-format)

- Open bash console and run **<PATH_TO_QUARTUS_ROOT_DIR>/nios2eds/nios2_command_shell.sh**

```
$ <PATH_TO_QUARTUS_ROOT_DIR/nios2eds/nios2_command_shell.sh
------------------------------------------------
Altera Nios2 Command Shell [GCC 4]

Version 13.0sp1, Build 232
------------------------------------------------
```

- Run sof2flash programm with arguments below

```
sof2flash --input=<PATH_TO_A5_SOF>/a5_sdk.sof --output=./a5_hw.flash
--offset=0x01640000 --pfl --optionbit=0x00018000 --programmingmode=FPP
```

- Wait for the generation to complete

```
Info: *******************************************************************
Info: Running Quartus II 32-bit Convert_programming_file
    Info: Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Full Version
    Info: Copyright (C) 1991-2013 Altera Corporation. All rights reserved.
    Info: Your use of Altera Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Altera Program License
    Info: Subscription Agreement, Altera MegaCore Function License
    Info: Agreement, or other applicable license agreement, including,
    Info: without limitation, that your use is for the sole purpose of
    Info: programming logic devices manufactured by Altera and sold by
    Info: Altera or its authorized distributors.  Please refer to the
    Info: applicable agreement for further details.
    Info: Processing started: Thu Nov 23 15:49:43 2017
Info: Command: quartus_cpf -c a5_hw.cof
Info (210033): Memory Map File a5_hw.map contains memory usage information for file
a5_hw.pof
Info: Quartus II 32-bit Convert_programming_file was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 162 megabytes
    Info: Processing ended: Thu Nov 23 15:49:59 2017
    Info: Elapsed time: 00:00:16
    Info: Total CPU time (on all processors): 00:00:16

Info: *******************************************************************
Info: Running Quartus II 32-bit Convert_programming_file
    Info: Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Full Version
    Info: Copyright (C) 1991-2013 Altera Corporation. All rights reserved.
    Info: Your use of Altera Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Altera Program License
    Info: Subscription Agreement, Altera MegaCore Function License
    Info: Agreement, or other applicable license agreement, including,
    Info: without limitation, that your use is for the sole purpose of
    Info: programming logic devices manufactured by Altera and sold by
    Info: Altera or its authorized distributors.  Please refer to the
    Info: applicable agreement for further details.
    Info: Processing started: Thu Nov 23 15:49:59 2017
Info: Command: quartus_cpf -c a5_hw.pof a5_hw.hexout
Info: Quartus II 32-bit Convert_programming_file was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 126 megabytes
    Info: Processing ended: Thu Nov 23 15:50:26 2017
    Info: Elapsed time: 00:00:27
    Info: Total CPU time (on all processors): 00:00:27


Extracting Option bits SREC
Extracting FPGA Image SREC
Deleting intermediate files
```

- Flash memory image (a5_hw.flash) is generated in the current folder
  Flash update steps described in section **3**

8.3.3. SDK-specific pins assignment in FPGA-project

Most of the pins assignments of the SDK project "inherit" from the basic design:
Arria V GX Starter Board Reference Manual

SDK-specific connection pins are used for interfaces UART0 and OpenOCD/JTAG. The purpose of these pins is shown below:

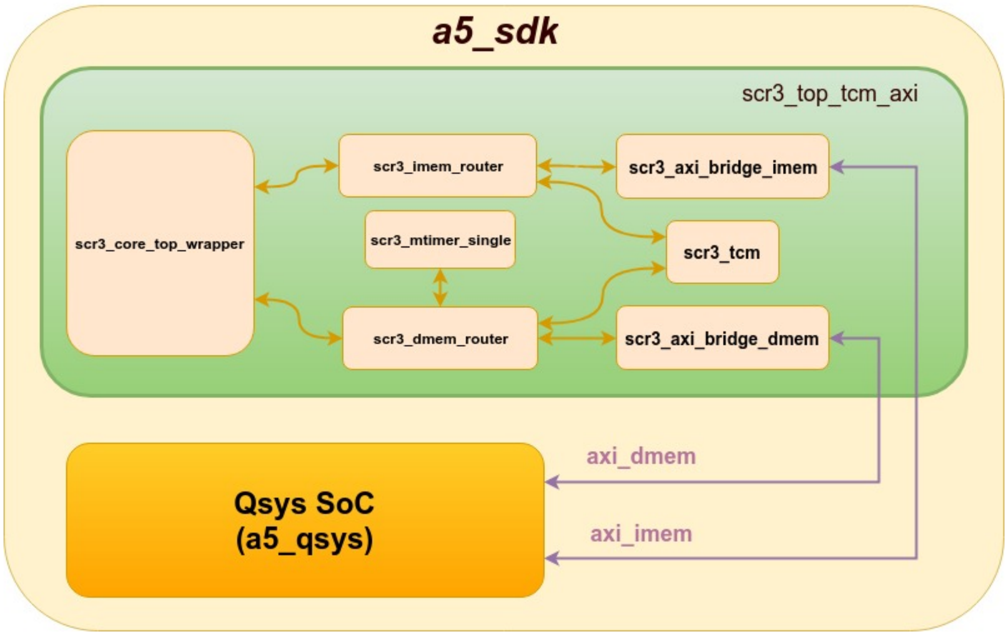
Table 3. SDK-specific pins assignment

FPGA-pin	Port name	I/O Standard	Descrpition
PIN_AJ10	uart0_rxd	2.5V	Input UART RXD
PIN_AH10	uart0_txd	2.5V	Output UART TXD
PIN_AJ6	jtag_trst_n	2.5V	Input JTAG TRSTn
PIN_AH6	jtag_tdi	2.5V	Input JTAG TDI
PIN_AC6	jtag_tms	2.5V	Input JTAG TMS
PIN_AC7	jtag_tck	2.5V	Input JTAG TCK
PIN_AM3	jtag_tdo	2.5V	Inout JTAG TDO
PIN_AD9	jtag_vcc[0]	2.5V	Output JTAG VCC
PIN_AM11	jtag_vcc[1]	2.5V	Output JTAG VCC
PIN_AL12	jtag_gnd[0]	2.5V	Output JTAG GND
PIN_AK12	jtag_gnd[1]	2.5V	Output JTAG GND
PIN_AM13	jtag_gnd[2]	2.5V	Output JTAG GND
PIN_AL13	jtag_gnd[3]	2.5V	Output JTAG GND
PIN_AH12	jtag_gnd[4]	2.5V	Output JTAG GND
PIN_AG12	jtag_gnd[5]	2.5V	Output JTAG GND
PIN_AJ13	jtag_gnd[6]	2.5V	Output JTAG GND
PIN_AH13	jtag_gnd[7]	2.5V	Output JTAG GND

8.4. SCR1 SDK FPGA-project functional description

8.4.1. Common project structure

The SDK project is configured and ready to be built immediately from the repository. The project contains the following main modules:



Modules description:

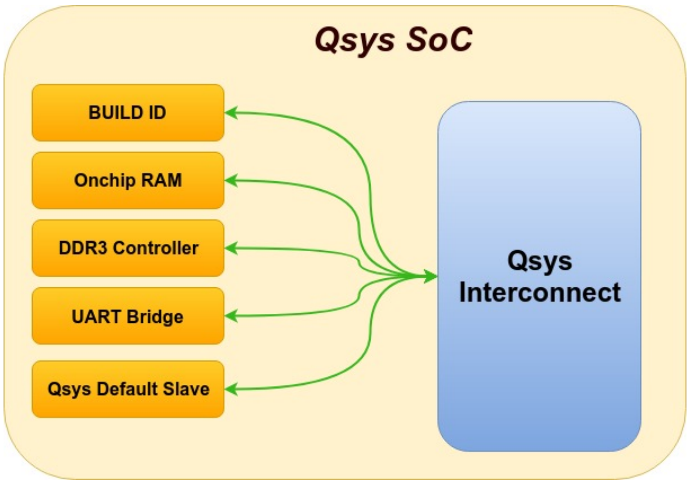
- **SCR1-core** - (supplied as an SystemVerilog RTL, is available from the repository)
- **Two data routers** - (imem_route/dmem_router - instruction/data transfers, supplied as an SystemVerilog RTL, is available from the repository)
- **Two AXI bridges** - (axi_imem/axi_dmem - instruction/data transfers, supplied as an SystemVerilog RTL, is available from the repository)
- **Timer block** - (external timer block, supplied as an SystemVerilog RTL, is available from the repository)
- **scr1_tcm** - (Tightly Coupled Memory (TCM), supplied as an SystemVerilog RTL, is available from the repository)
- **UART 16550** - (Opencores UART 16550 IP, supplied as an Verilog RTL, is available from the repository)
- **Qsys SoC block** - (Qsys component, containing the generated IP-components)

8.4.2. Qsys SoC module structure

Qsys SoC module consists of:

- BUILD ID
- Onchip RAM
- DDR3 controller

- UART Bridge
- Qsys Default slave



8.4.3. Description of the blocks used in the SDK project

8.4.3.1. SCR1-core

Syntacore RISC-V core.
The core is supplied as SystemVerilog RTL sources.
A detailed description of the external interfaces of the core and other details are described in the document "SCR1 External Architecture Specification".

8.4.3.2. AXI bridge

The AXI bridge module is used to convert the internal interface of the core to the external interface that is compatible with AXI4 for external memory access. The module is provided in the source code (RTL) and contains an intermediate buffer for the external interface requests.

8.4.3.3. Opencores UART 16550 IP

There is a 16550 compatible (mostly) UART core. The bus interface is WISHBONE SoC bus Rev. B.
Features all the standard options of the 16550 UART: FIFO based operation, interrupt requests and other.
UART 16550 core project

8.4.4. Description of the IP-components of the module Qsys SOC

Detailed description of Altera’s common Qsys components used in the project:

- Embedded Peripheral IP User Guide
- Qsys System Design Tutorial
- Qsys Interconnect

8.4.4.1. BUILD ID

PIO-block contains the project build date parameter, which is available for reading by the

processor.
The parameter FPGA_A5_BUILD_ID is defined in the file scr1_arch_custom.svh.
Component base address - 0xFF000000.

8.4.4.2. Onchip RAM

Internal bootload memory with bootloader code resides in the FPGA.
Memory size - 64KB.
Memory base address - 0xFFFF0000.
Memory initialization data is supplied in a5_sdk.hex (hex format). The code and instructions for building the bootloader are available from the current repository.

Further details:
http://www.altera.com/literature/hb/qts/qts_qii54006.pdf

8.4.4.3. DDR3 Controller

The component is used to configure and manage external DDR3 memory using the Hard-IP block for the FPGA family Arria V. The component is configured to work with an external memory of 256MB at DDR3 266(533) MHz.
Component base address - 0x00000000.

Further details:
http://www.altera.com/support/literature/lit-external-memory-interface.html

8.4.4.4. UART Bridge

Altera avalon bridge for the Opencores UART 16550 IP.
Component base address - 0xFF010000.

8.4.4.5. Qsys Default slave

The slave responder component by "default". The main function is to generate an error status for any transactions in the unused ranges of the addresses of the Qsys SoC.

9. Appendix A. Software build instructions

This build guide describes how to build software provided as a part of the SCR1 SDK.

9.1. SCR bootloader

9.1.1. Getting the sources

```
$ git clone git@github.com:syntacore/sc-bl.git
```

9.1.2. Building SCR bootloader

Follow the instructions in sc-bl/README.md to build bootloader for target plaforms ('scbl.hex' for Terasic DE10-Lite, 'scbl.mem' for Digilent Arty).

9.2. Zephyr OS

9.2.1. Getting the sources

```
$ git clone git@github.com:syntacore/zephyr.git
```

9.2.2. Building Zephyr OS

Follow the instructions in https://www.zephyrproject.org/doc/getting_started/getting_started.html and zephyr/README.md to build Zephyr OS image for target plaform.

9.3. SCR1 OpenOCD

9.3.1. Getting the sources

```
$ git clone -b syntacore https://github.com/syntacore/openocd
```

9.3.2. Building and using OpenOCD

Please, refer to the Syntacore OpenOCD wiki page for instructions: https://github.com/syntacore/openocd/wiki/OpenOCD-for-sc_riscv32