SCR1 User Manual

Syntacore, scr1@syntacore.com

Version 1.1.2, 2020-11-13

Table of Contents

Revision history

Revision	Date	Description
1.0.0	2018-05-07	Initial version
1.0.1	2018-09-19	RTL configurations and sim script update
1.0.2	2018-10-09	Updated MIMPID
1.0.3	2019-03-19	Updated to comply with MIMPID=0x19031802
1.0.4	2019-04-11	Updated to comply with MIMPID=0x19040301
1.0.5	2019-05-07	Updated simulation environment
1.0.6	2019-08-30	Updated MIMPID=0x19083000
1.0.7	2019-10-28	Updated chapter 'Test subset'
1.1.0	2019-12-13	New SCR1 cluster diagram. Detailed description on filelists. Updated setup procedure for simulation. More information on SCR1 SDK repo.
1.1.1	2020-07-15	Updated Compliance tests, added TCM option
1.1.2	2020-11-13	Updated to comply with MIMPID=0x20111300

1. SCR1 overview

SCR1 is an open-source and free to use RISC-V compatible MCU-class core, designed and maintained by Syntacore. See the LICENSE file in the root directory for details.

1.1. Version of SCR1

This document is relevant for SCR1 core with MIMPID value of 0x20111300.

1.2. Key features

- Open sourced under SHL-license (see LICENSE file) - unrestricted commercial use allowed
- RV32I or RV32E ISA base with optional RVM and RVC standard extensions
- Machine privilege mode only
- 2 to 4 stage pipeline
- Optional Integrated Programmable Interrupt Controller with 16 IRQ lines
- Optional RISC-V Debug subsystem with JTAG interface
- Optional on-chip Tightly-Coupled Memory
- 32-bit AXI4/AHB-Lite external interface
- Written in SystemVerilog
- Optimized for area and power consumption
- 3 predefined recommended configurations
- A number of fine-tuning options for custom configuration
- Verification suite provided
- Extensive documentation

For more information on core architecture see SCR1 External Architecture Specification (EAS).

2. Codebase overview

2.1. SCR1 repository content

Table 1: Directories and content

Folder	Description
dependencies	**Dependent submodules**
riscv-tests	Common source files for RISC-V ISA tests
riscv-compliance	Common source files for RISC-V Compliance tests
coremark	Common source files for EEMBC's CoreMark® benchmark
docs	**SCR1 documentation**
scr1_eas.pdf	SCR1 External Architecture Specification
scr1_um.pdf	SCR1 User Manual
sim	**Tests and scripts for simulation**
tests/common	Common source files for tests
tests/riscv_isa	RISC-V ISA tests platform specific source files
tests/riscv_compliance	RISC-V Compliance platform specific source files
tests/benchmarks/dhrystone21	Dhrystone 2.1 benchmark source files
tests/benchmarks/coremark	EEMBC's CoreMark® benchmark platform specific source files
tests/isr_sample	Sample program "Interrupt Service Routine"
tests/hello	Sample program "Hello"
verilator_wrap	Wrappers for Verilator simulation
src	**SCR1 RTL source and testbench files**
includes	Header files
core	Core top source files
top	Cluster source files
tb	Testbench files

2.2. SCR1 RTL source and testbench files

SCR1 source file lists of SCR1 can be found in *./src*:

- **core.files** - all synthesized file sources of the SCR1 core
- **ahb_top.files** - synthesized file sources of AHB cluster
- **axi_top.files** - synthesized file sources of AXI cluster
- **ahb_tb.files** - testbench file sources for AHB cluster (for simulation only)

- **axi_tb.files** - testbench file sources for AXI cluster (for simulation only)

Library with header files to include is *./src/includes/*

Below is a complete list of all source files.

Table 2: SCR1 RTL source and testbench files

Path	Description
SCR1 header files	
includes/scr1_ahb.svh	AHB header file
includes/scr1_arch_description.svh	Architecture description file
includes/scr1_arch_types.svh	Pipeline types description file
includes/scr1_csr.svh	CSR mapping/description file
includes/scr1_dm.svh	DM header file
includes/scr1_hdu.svh	HDU header file
includes/scr1_ipic.svh	IPIC header file
includes/scr1_memif.svh	Memory interface definitions file
includes/scr1_riscv_isa_decoding.svh	RISC-V ISA definitions file
includes/scr1_scu.svh	SCU header file
includes/scr1_search_ms1.svh	Most significant one search function
includes/scr1_tapc.svh	TAPC header file
includes/scr1_tdu.svh	TM header file
SCR1 core source files	
core/pipeline/scr1_ipic.sv	Integrated Programmable Interrupt Controller (IPIC)
core/pipeline/scr1_pipe_csr.sv	Control Status Registers (CSR)
core/pipeline/scr1_pipe_exu.sv	Execution Unit (EXU)
core/pipeline/scr1_pipe_hdu.sv	Hart Debug Unit (HDU)
core/pipeline/scr1_pipe_ialu.sv	Integer Arithmetic Logic Unit (IALU)
core/pipeline/scr1_pipe_idu.sv	Instruction Decoder Unit (IDU)
core/pipeline/scr1_pipe_ifu.sv	Instruction Fetch Unit (IFU)
core/pipeline/scr1_pipe_lsu.sv	Load/Store Unit (LSU)
core/pipeline/scr1_pipe_mprf.sv	Multi Port Register File (MPRF)
core/pipeline/scr1_pipe_tdu.sv	Trigger Debug Unit (TDU)
core/pipeline/scr1_pipe_top.sv	SCR1 pipeline top
core/pipeline/scr1_tracelog.sv	Core tracelog module (for simulation only)
core/primitives/scr1_cg.sv	SCR1 clock gate primitive
core/primitives/scr1_reset_cells.sv	SCR1 reset logic primitives
core/scr1_clk_ctrl.sv	SCR1 clock control

Path	Description
core/scr1_core_top.sv	SCR1 core top
core/scr1_dm.sv	Debug Module (DM)
core/scr1_dmi.sv	Debug Module Interface (DMI)
core/scr1_scu.sv	System Control Unit
core/scr1_tapc.sv	TAP Controller (TAPC)
core/scr1_tapc_shift_reg.sv	TAPC shift register
core/scr1_tapc_synchronizer.sv	TAPC clock domain crossing synchronizer
SCR1 top cluster source files	
top/scr1_dmem_ahb.sv	Data memory AHB bridge
top/scr1_dmem_router.sv	Data memory router
top/scr1_dp_memory.sv	Dual-port synchronous memory with byte enable inputs
top/scr1_imem_ahb.sv	Instruction memory AHB bridge
top/scr1_imem_router.sv	Instruction memory router
top/scr1_mem_axi.sv	Memory AXI bridge
top/scr1_tcm.sv	Tightly-Coupled Memory (TCM)
top/scr1_timer.sv	Memory-mapped Timer
top/scr1_top_ahb.sv	SCR1 AHB top
top/scr1_top_axi.sv	SCR1 AXI top
Testbench files	
tb/scr1_memory_tb_ahb.sv	AHB memory testbench
tb/scr1_memory_tb_axi.sv	AXI memory testbench
tb/scr1_top_tb_ahb.sv	SCR1 top testbench AHB
tb/scr1_top_tb_axi.sv	SCR1 top testbench AXI
tb/scr1_top_tb_runtests.sv	Testbench run tests

3. Core configurations

3.1. Core and device identifiers

The table below shows SCR1 core and device identifiers.

Table 3: SCR1 core and device identifiers

Identifier name	Description
MIMPID	SCR1 core implementation ID to read from corresponding CSR. The number uniquely identifies the version of the SCR1 core RTL.
MARCHID	SCR1 core architecture ID to read from corresponding CSR. The number identifies the SCR1 core from other RISC-V cores. Hardwired to 0x00000008
MVENDORID	SCR1 core vendor ID to read from corresponding CSR. For commercial manufacturing purposes, please overwrite this field to your JEDEC Standard Manufacturer's ID Code. Default value is 0x00000000
TAP_IDCODE	IDCODE to read from the corresponding TAPC register via JTAG. For commercial manufacturing purposes, please overwrite this field to your JEDEC Standard Manufacturer's ID Code + part number and version. Default value is 0xDEB11001 (not JEDEC)
BUILD_ID	Device build ID. The number must be set in a external arch_custom.svh file for a specific device build (e.g. for FPGA build in SCR1-SDK). Default value for simulation = MIMPID

3.2. Recommended configurations

The table below shows three recommended SCR1 configurations for typical use cases. These configurations can be easily enabled in **scr1_arch_description.svh** file, section *"RECOMMENDED CORE ARCHITECTURE CONFIGURATIONS"*. To select a configuration, uncomment the only relevant *define* from the list.

Table 4: SCR1 recommended configurations

Options	SCR1_CFG_RV32EC_MIN	SCR1_CFG_RV32IC_BASE	SCR1_CFG_RV32IMC_MAX
Instruction set	RV32EC	RV32IC	RV32IMC
Pipeline stages	2	3	4
Number of GPRs	16	32	32
Hardware multiplier	-	-	+
Fast 1-cycle multiplier	-	-	+
Compressed instructions	+	+	+
MTVEC base address writable bits	0	16	26
MTVEC mode writable	-	+	+
External IRQ lines	1	16	16

Options	SCR1_CFG_RV32EC_MIN	SCR1_CFG_RV32IC_BASE	SCR1_CFG_RV32IMC_MAX
Debug subsystem	-	+	+
Number of hardware triggers	0	2	4
TCM	+	+	+

3.3. Fine-tuning options for custom configuration

SCR1 has a number of fine-tuning options for custom configuration described in the table below. To make your own design of these options, you need to edit **scr1_arch_description.svh** file:

- undefine all recommended configurations in the section *"RECOMMENDED CORE ARCHITECTURE CONFIGURATIONS"* to enable custom configuration,
- select all the necessary options in section *"CUSTOM CORE ARCHITECTURE CONFIGURATION"*:
  - to disable/enable an options - comment/uncomment the corresponding *define*,
  - for numeric parameter - change it's value.

Table 5: SCR1 configurable options

Name	Description
RISC-V ISA options	
SCR1_RVE_EXT	Enable RV32E base integer instruction set, otherwise RV32I will be used
SCR1_RVM_EXT	Enable standard extension "M" for integer hardware multiplier and divider
SCR1_RVC_EXT	Enable standard extension "C" for compressed instructions
SCR1_MTVEC_BASE_WR_BITS	Number of writable bits in MTVEC.base field
SCR1_MTVEC_MODE_EN	Enable writable MTVEC.mode field to allow vectored irq mode, otherwise only direct mode is possible
Core pipeline options (power-performance-area optimization)	
SCR1_NO_DEC_STAGE	Disable register between IFU and IDU
SCR1_NO_EXE_STAGE	Disable register between IDU and EXU
SCR1_NEW_PC_REG	Enable register in IFU for New PC value
SCR1_FAST_MUL	Enable fast one-cycle multiplication, otherwise multiplication takes 32 cycles
SCR1_CLKCTRL_EN	Enable global clock gating
SCR1_MPRF_RST_EN	Enable reset for MPRF
SCR1_MCOUNTEN_EN	Enable custom MCOUNTEN CSR for counter control
Uncore options	
SCR1_DBG_EN	Enable Debug Subsystem (TAPC, DM, SCU, HDU)
SCR1_TDU_EN	Enable Trigger Debug Unit (hardware breakpoints)

Name	Description
SCR1_TDU_TRIG_NUM	Number of hardware triggers
SCR1_TDU_ICOUNT_EN	Enable hardware triggers on instruction counter
SCR1_IPIC_EN	Enable Integrated Programmable Interrupt Controller
SCR1_IPIC_SYNC_EN	Enable 2-stage input synchronizer for IRQ lines
SCR1_TCM_EN	Enable Tightly-Coupled Memory, default size is 64K

NOTE For synthesis if enable SCR1_CLKCTRL_EN code in **scr1_cg.sv** should be replaced with implementation-specific clock gate.

3.4. Core integration options

SCR1 has a number options for integration into upper-level design. This options can be changed in **scr1_arch_description.svh** file, section *"CORE INTEGRATION OPTIONS"*:

- to disable/enable an options - comment/uncomment the corresponding *define*,
- for numeric parameter - change it's value.

Some options can be defined in the external file **scr1_arch_custom.svh** which is not presented in the SCR1 repo, but can be used in upper-level project (e.g. open SCR1-SDK project and any other custom FPGA, ASIC or SoC projects).

Table 6: SCR1 integration options

Name	Description
Memory bridges bypass options	
SCR1_IMEM_AHB_IN_BP	Enable bypass on instruction memory AHB bridge inputs
SCR1_IMEM_AHB_OUT_BP	Enable bypass on instruction memory AHB bridge outputs
SCR1_DMEM_AHB_IN_BP	Enable bypass on data memory AHB bridge inputs
SCR1_DMEM_AHB_OUT_BP	Enable bypass on data memory AHB bridge outputs
SCR1_IMEM_AXI_REQ_BP	Enable bypass on instruction memory AXI bridge request
SCR1_IMEM_AXI_RESP_BP	Enable bypass on instruction memory AXI bridge response
SCR1_DMEM_AXI_REQ_BP	Enable bypass on data memory AXI bridge request
SCR1_DMEM_AXI_RESP_BP	Enable bypass on data memory AXI bridge response
Address constants	
SCR1_ARCH_RST_VECTOR	Reset vector value (start address after reset) (default 0x200)
SCR1_ARCH_MTVEC_BASE	MTVEC.base field reset value, or constant value for MTVEC.base bits that are hardwired (default 0x1C0)

Name	Description
SCR1_TCM_ADDR_MASK	Set TCM mask and size; size in bytes is two's complement of the mask value (default 0xFFFF0000)
SCR1_TCM_ADDR_PATTERN	Set TCM address match pattern (default 0x00480000)
SCR1_TIMER_ADDR_MASK	Set timer mask (default 0xFFFFFFE0)
SCR1_TIMER_ADDR_PATTERN	Set timer address match pattern (default 0x00490000)
Target platform (enables target-specific constructs)	
SCR1_TRGT_FPGA_INTEL	Target platform is Intel FPGAs
SCR1_TRGT_FPGA_INTEL_MAX10	Target platform is Intel MAX 10 FPGAs (used in the SCR1-SDK project)
SCR1_TRGT_FPGA_INTEL_ARRIAV	Target platform is Intel Arria V FPGAs (used in the SCR1-SDK project)
SCR1_TRGT_FPGA_XILINX	Target platform is Xilinx FPGAs (used in the SCR1-SDK project)
SCR1_TRGT_ASIC	Target platform is ASIC

3.5. Options for simulation

The parameters below are used for simulation only to enable the simulation code and set up the testbench. These options can be changed in **scr1_arch_description.svh** file, section *"SIMULATION OPTIONS"*:

- to disable/enable an options - comment/uncomment the corresponding *define*,
- for numeric parameter - change it's value.

Table 7: SCR1 simulation options

Name	Description
Simulation options	
SCR1_TRGT_SIMULATION	Enable simulation code (automatically defined by root makefile) (see Simulation code)
SCR1_TRACE_LOG_EN	Enable tracelog (see Tracelog)
Addresses used in testbench (see Testbench description)	
SCR1_SIM_EXIT_ADDR	Write this address to exit the simulation (default 0x000000F8)
SCR1_SIM_PRINT_ADDR	Write this address to print a symbol in console (default 0xF0000000)
SCR1_SIM_EXT_IRQ_ADDR	Write this address to generate external interrupts (default 0xF0000100)
SCR1_SIM_SOFT_IRQ_ADDR	Write this address to generate software interrupt (default 0xF0000200)

4. Simulation environment

The project contains testbenches, test sources and scripts to quickly start the SCR1 simulation. Before starting the simulation, make sure you have:

- installed RISC-V GCC toolchain,
- installed one of the supported simulators,
- initialized submodules with test sources.

4.1. Requirements

4.1.1. Operating system

GCC toolchain and make-scripts are supported by most popular Linux-like operating systems.

To run from Windows you can use an additional compatibility layer, such as WSL or Cygwin.

4.1.2. RISC-V GCC toolchain

RISC-V GCC toolchain is required to compile the software. You can use pre-built binaries or build the toolchain from scratch.

4.1.2.1. Using pre-built binary tools

Pre-built RISC-V GCC toolchain with support for all SCR1 architectural configurations is available for download from http://syntacore.com/page/products/sw-tools.

1. Download the archive for your platform.
2. Extract the archive to preferred directory `<GCC_INSTALL_PATH>`.
3. Add the `<GCC_INSTALL_PATH>/bin` folder to the $PATH environment variable:

```
export PATH=<GCC_INSTALL_PATH>/bin:$PATH
```

4.1.2.2. Building tools from source

You can build the RISC-V GCC toolchain from sources, stored in official repo https://github.com/riscv/riscv-gnu-toolchain

Instructions on how to prepare and build the toolchain can be found on https://github.com/riscv/riscv-gnu-toolchain/blob/master/README.md

We recommend using the multilib compiler. Please note that RV32IC, RV32E, RV32EM, RV32EMC, RV32EC architectural configurations are not included in the compiler by default. If you plan to use them, you will need to include the appropriate libraries by yourself before building.

After the building, be sure to add the `<GCC_INSTALL_PATH>/bin` folder to the $PATH environment

variable

4.1.3. HDL simulators

Currently supported simulators:

- Verilator (last verified version: v4.102)
- Intel ModelSim (last verified version: INTEL FPGA STARTER EDITION vsim 2020.1_3)
- Mentor Graphics ModelSim (last verified version: Modelsim PE Student Edition 10.4a)
- Synopsys VCS (last verified version: vcs-mx_vL-2016.06)
- Cadence NCSim

Please note that RTL simulator executables should be in your $PATH variable.

4.1.4. Tests preparation

The simulation package includes the following tests:

- **hello** - "Hello" sample program
- **isr_sample** - "Interrupt Service Routine" sample program
- **riscv_isa** - RISC-V ISA tests (submodule)
- **riscv_compliance** - RISC-V Compliance tests (submodule)
- **dhrystone21** - Dhrystone 2.1 benchmark
- **coremark** - EEMBC's CoreMark® benchmark (submodule)

After the main SCR1 repository has been cloned execute the following command:

```
git submodule update --init --recursive
```

This command will initialized submodules with test sources.

4.2. Running simulation

To build RTL, compile and run tests from the repo root folder you have to call Makefile. By default, you may simply call Makefile without any parameters:

```
make
```

In this case simulation will run on Verilator with following parameters: `CFG=MAX BUS=AHB TRACE=0 TARGETS="hello isr_sample riscv_isa riscv_compliance dhrystone21 coremark"`.

Makefile supports:

- choice of simulator - `run_<SIMULATOR>`,

- architecture setup - `CFG, BUS, ARCH, VECT_IRQ, IPIC, TCM`,
- tests subset to run - `TARGETS`
- enabling tracelog - `TRACE`
- and any additional options to pass to the simulator - `SIM_BUILD_OPTS`.

Example:

```
    make run_vcs CFG=CUSTOM BUS=AXI ARCH=I VECT_IRQ=1 IPIC=1 TCM=0 TARGETS="hello
isr_sample" TRACE=1 SIM_BUILD_OPTS="-gui"
```

Build and run parameters can be configured in the `./Makefile`.

After all the tests have finished, the results can be found in `build/<SIM_CFG>/test_results.txt`.

IMPORTANT	To ensure correct rebuild, please call **make clean** between simulation runs.

4.2.1. Simulator selection

You may specify one of supported simulators `run_<SIMULATOR> = <run_vcs, run_modelsim, run_ncsim, run_verilator, run_verilator_wf>`:

```
    make run_modelsim
```

Simulator run:

- **run_verilator** - Verilator (default)
- **run_verilator_wf** - Verilator with waveforms generation
- **run_modelsim** - ModelSim by Mentor Graphics or Intel
- **run_vcs** - Synopsys VCS
- **run_ncsim** - Cadence NCSim

For the run_verilator_wf option, a waveform is generated for all tests performed and saved in `./build/<SIM_CFG>/simx.vcd`. The file can be opened by some waveform viewer, such as GTKWave.

4.2.2. Architectural configuration

You may specify configuration `CFG = <MAX, BASE, MIN, CUSTOM>` and external interface `BUS = <AHB, AXI>`:

```
    make CFG=BASE BUS=AXI
```

Configurations expand as follows:

- **MAX** - sets predefined configuration SCR1_CFG_RV32IMC_MAX (default)

- **BASE** - sets predefined configuration SCR1_CFG_RV32IC_BASE
- **MIN** - sets predefined configuration SCR1_CFG_RV32EC_MIN
- **CUSTOM** - could be used for any other custom configurations

For all predefined configurations, other architectural parameters are automatically set to a deterministic state, both for compiling tests and SCR1 RTL.

For **CUSTOM** configurations, you can specify additional parameters:

- **ARCH** = **<IMC, IC, IM, I, EMC, EM, EC, E>** - RISC-V instruction set architecture. The parameter defines the RISC-V instruction set architecture for compiling tests (automatically used by the RISC-V toolchain): RV32I or RV32E base + optional standard extensions M and C. RTL options SCR1_RVE_EXT, SCR1_RVM_EXT and SCR1_RVC_EXT must be defined accordingly.
- **VECT_IRQ** = **<0, 1>** - vectored mode to handle interrupts, otherwise direct mode is used. The definition of the parameter VECT_IRQ is used in the test "isr_sample" to show various interrupt call and handling scenarios. RTL option SCR1_MTVEC_MODE_EN must be defined for vectored mode.
- **IPIC** = **<0, 1>** - using Integrated Programmable Interrupt Controller. The definition of the parameter IPIC is used in the test "isr_sample" to show various interrupt call and handling scenarios. RTL option SCR1_IPIC_EN must be defined accordingly.
- **TCM** = **<0, 1>** - using Tightly Coupled Memory. Setting TCM option to 1 defines some tests to be executed from Tightly Coupled Memory instead of external testbench memory. RTL option SCR1_TCM_EN must be defined accordingly

IMPORTANT	Set of additional parameters for a CUSTOM configuration doesn't enable the SCR1 RTL parameters. Please, don't forget to manually set the corresponding parameters in the file `./src/includes/scr1_arch_description.svh`.

NOTE	Additional parameters cannot be used for predefined configurations as they are already hardcoded.

Example:

```
make CFG=CUSTOM ARCH=I VECT_IRQ=1 IPIC=1 TCM=0
```

4.3. Targets

You can specify a test subset to run in a simulation:

- **TARGETS** = **<hello, isr_sample, riscv_isa, riscv_compliance, dhrystone21, coremark>**

To select only one target from the list, specify its name, for example:

```
make TARGETS=hello
```

To select multiple targets, list them in quotation marks separated by spaces, for example:

```
make TARGETS="dhrystone21 coremark"
```

Some of the tests depend on the selected architecture and therefore can not be used for all core configurations (these are skipped automatically).

To select individual tests from a collection, you need:

- For the **riscv_isa** collection, go to **./sim/tests/riscv_isa/rv32_tests.inc** and list the required tests in the **rv32_isa_tests**.
- For the **riscv_compliance** collection, go to **./sim/tests/riscv_compliance/Makefile** and list the required tests in the **compliance_set** (specify the full path to each test).

4.4. Simulation code

You can add useful information about the simulation process: assertions, tracelog and instruction statistics. The SCR1_TRGT_SIMULATION parameter must be defined in `./src/includes/scr1_arch_description.svh` section "SIMULATION OPTIONS" to enable all simulation code. This parameter is automatically enabled when you run the make-script.

4.4.1. Tracelog

During the simulation, the following information can be written to a special file **tracelog_core_N.log** in build directory:

- RTL_ID value
- Core reset events and time of their occurrence
- MPRF and CSR registers update information in the following format:

```
Time | Event | Curr_PC | Instr | Next_PC | Reg | Value
```

The following abbreviations of events are used:

- N - no event (regular register value update or cycle without updates)
- E - exception
- I - interrupt
- W - wakeup

Parameter SCR1_TRACE_LOG_EN and SCR1_TRGT_SIMULATION must be defined in **src/includes/scr1_arch_description.svh** to enable tracelog. When using make-script, you can pass parameter `TRACE=1` to automatically enable tracelog generation for all selected tests.

4.5. Testbench description

SCR1 testbench consists of top level module and external memory. Two testbench configurations are available depending on the memory interface used: AXI or AHB.

The desired configuration can be chosen by specifying the `BUS=<AHB, AXI>` option of make command (for the full command refer to [Running simulations]). The default value is `BUS=AHB`.

The list of files for both configurations is provided in the table below.

Table 8: SCR1 Testbench files

Path	Description
AXI Testbench	
tb/scr1_memory_tb_axi.sv	AXI memory testbench
tb/scr1_top_tb_axi.sv	SCR1 top testbench AXI
AHB Testbench	
tb/scr1_memory_tb_ahb.sv	AHB memory testbench
tb/scr1_top_tb_ahb.sv	SCR1 top testbench AHB

Both testbench memories have the size of 1024 kB.

NOTE	If TCM is enabled its memory address ranges are cut from external memory address ranges.

Testbench memories provide the mechanism for generating interrupts and printing characters in simulation console by writing data to the specific address. Also loading Program Counter with Simulation Exit address value terminates the simulation. Defines for such addresses are located in "SIMULATION OPTIONS" section of **scr1_arch_description.svh**. Default addresses map is shown in Table 1.

Table 9: SCR1 Interrupts and Simulation Control Default Memory Map

Address	Description
0xF0000100	External IRQs
0xF0000200	Software IRQ
0xF0000000	Print Character
0x000000F8	Simulation Exit

External 1-pin IRQ and IPIC IRQ lines share the same address. Make sure you are using the correct width, depending on whether IPIC is enabled. External 1-pin IRQ (if IPIC disabled) and Soft IRQ values should be placed in bit 0. IRQ Lines values (if IPIC enabled) uses 16 least significant bits of write data.

NOTE	If you need to use addresses map other than default make sure both RTL and test program use the same map.

5. SDK information

Open-source FPGA-based SDKs are available at the https://github.com/syntacore/scr1-sdk.

Repo contains:

- Pre-build images and open designs for several standard FPGAs boards:
  - Digilent Arty (Xilinx)
  - Digilent Nexys 4 DDR (Xilinx)
  - Arria V GX Starter (Intel)
  - Terasic DE10-Lite (Intel)
- Software package:
  - Bootloader
  - Zephyr RTOS
  - Tests and SW samples
- User Guides for SDKs and tools

6. Support

For more information on SCR1 core, please write to scr1@syntacore.com.