SCR1 User Manual

Syntacore, scr1@syntacore.com

Version 1.1.4, 2022-01-13

SCR1 用户手册

Syntacore， scr1@syntacore.com

版本 1.1.4、2022-01-13

Table of Contents

目录

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
1.1.3	2021-02-08	Updated to comply with MIMPID=0x21020800
1.1.4	2022-01-13	MIMPID=0x22011200

修订历史记录

校订	日期描述	
1.0.0	SDK 信息 2018-05-07 初始版本	
1.0.1	2018-09-19 RTL 配置和 sim 脚本更新	
1.0.2	2018-10-09 更新了 MIMPID	
1.0.3	2019 年 3 月 19 日更新以符合 MIMPID=0x19031802	
1.0.4	2019 年 4 月 11 日更新以符合 MIMPID=0x19040301	
1.0.5	2019-05-07 更新了模拟环境	
1.0.6	2019-08-30 更新了 MIMPID=0x19083000	
1.0.7	2019-10-28 更新了“测试子集”一章	
1.1.0	2019-12-13 新的 SCR1 集群图。	文件列表的详细说明。 更新了仿真的设置过程。 有关 SCR1 SDK 存储库的更多信息。
1.1.1	2020-07-15 更新了合规性测试，添加了 TCM 选项	
1.1.2	2020 年 11 月 13 日更新以符合 MIMPID=0x20111300	
1.1.3	2021 年 2 月 8 日更新以符合 MIMPID=0x21020800	
1.1.4	2022 年 1 月 13 日 MIMPID=0x22011200	

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 0x22011200.

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).

1. SCR1 概述

SCR1 是一个开源且免费使用的兼容 RISC-V 的 MCU 级内核，由 Syntacore 设计和维护。有关详细信息，请参阅根目录中的 LICENSE 文件。

1.1. SCR1 的版本

本文档与 MIMPID 值为 0x22011200 的 SCR1 内核相关。

1.2. 主要特点

- 在 SHL 许可下开源（参见 LICENSE 文件） - 允许不受限制的商业用途
- RV32I 或 RV32E ISA 底座，带有可选的 RVM 和 RVC 标准扩展
- 仅限计算机特权模式
- 2 至 4 级管道
- 可选的集成可编程中断控制器，具有 16 条 IRQ 线
- 带 JTAG 接口的可选 RISC-V 调试子系统
- 可选的片上紧耦合存储器
- 32 位 AXI4/AHB-Lite 外部接口
- 以 SystemVerilog 编写
- 针对面积和功耗进行了优化
- 3 种预定义的推荐配置
- 用于自定义配置的多个微调选项
- 提供验证套件
- 广泛的文档

有关核心体系结构的详细信息，请参阅 SCR1 外部体系结构规范 （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)

2. 代码库概述

2.1. SCR1 存储库内容

表 1：目录和内容

文件夹	描述
依赖	依赖子模块
├── RISCV 测试	RISC-V ISA 测试的通用源文件
├── RISCV 合规	RISC-V 一致性测试的常见源文件
└── coremark	EEMBC 的 CoreMark® 基准测试的常见源文件
docs	SCR1 文档
├── scr1_eas.pdf	SCR1 外部架构规范
└── scr1_um.pdf	SCR1 用户手册
sim	用于模拟的测试和脚本
├── 检验/常见	测试的常见源文件
├── 测试/riscv_isa	RISC-V ISA 测试特定于平台的源文件
├── 测试/riscv_compliance	RISC-V 一致性测试平台特定的源文件
├── 测试/基准测试/dhrystone21	Dhrystone 2.1 基准测试源文件
├── 测试/基准测试/coremark	EEMBC 的 CoreMark® 基准测试平台特定源文件
├── 测试/isr_sample	示例程序 “Interrupt Service Routine”
├── tests/hello	示例程序 “Hello”
└── verilator_wrap	用于 Verilator 仿真的包装器
src	SCR1 RTL 源和测试平台文件
├── 包括	头文件
├── 芯	核心 top 源文件
├── 返回页首	群集源文件
└── 结核病	测试台文件

2.2. SCR1 RTL 源和测试台文件

SCR1 的 SCR1 源文件列表可以在 ./src 中找到：

- core.files - SCR1 内核的所有综合文件源
- ahb_top.files - AHB 集群的合成文件源
- axi_top.files - AXI 群集的合成文件源
- ahb_tb.files - AHB 集群的测试台文件源（仅用于仿真）

- **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

- axi_tb.files - AXI 群集的测试台文件源（仅用于仿真）

包含头文件的库是 ./src/includes/

以下是所有源文件的完整列表。

表 2：SCR1 RTL 源和测试平台文件

Path	描述
SCR1 头文件	
includes/scr1_ahb.svh	AHB 头文件
includes/scr1_arch_description.svh	架构描述文件
includes/scr1_arch_types.svh	管道类型描述文件
includes/scr1_csr.svh	CSR 映射/描述文件
includes/scr1_dm.svh	DM 头文件
includes/scr1_hdu.svh	HDU 头文件
includes/scr1_ipic.svh	IPIC 头文件
includes/scr1_memif.svh	内存接口定义文件
includes/scr1_riscv_isa_decoding.svh	RISC-V ISA 定义文件
includes/scr1_scu.svh	SCU 头文件
includes/scr1_search_ms1.svh	最重要 1 搜索功能
includes/scr1_tapc.svh	TAPC 头文件
includes/scr1_tdu.svh	TM 头文件
SCR1 核心源文件	
核心/管道/scr1_ipic.sv	集成可编程中断控制器 （IPIC）
核心/管道/scr1_pipe_csr.sv	控制状态寄存器 （CSR）
core/pipeline/scr1_pipe_exu.sv	执行单元 （EXU）
核心/管道/scr1_pipe_hdu.sv	Hart 调试单元 （HDU）
核心/管道/scr1_pipe_ialu.sv	整数算术逻辑单元 （IALU）
核心/管道/scr1_pipe_idu.sv	指令解码器单元 （IDU）
核心/管道/scr1_pipe_ifu.sv	指令获取单元 （IFU）
核心/管道/scr1_pipe_lsu.sv	加载/存储单元 （LSU）
核心/管道/scr1_pipe_mprf.sv	多端口寄存器文件 （MPRF）
核心/管道/scr1_pipe_tdu.sv	触发调试单元 （TDU）
核心/管道/scr1_pipe_top.sv	SCR1 管道顶部
core/pipeline/scr1_tracelog.sv	核心 tracelog 模块（仅用于模拟）
核心/基元/scr1_cg.sv	SCR1 时钟门原语
核心/基元/scr1_reset_cells.sv	SCR1 复位逻辑原语
核心/scr1_clk_ctrl.sv	SCR1 时钟控制

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

Path	描述
core/scr1_core_top.sv	SCR1 核心顶部
core/scr1_dm.sv	调试模块 （DM）
核心/scr1_dmi.sv	调试模块接口 （DMI）
core/scr1_scu.sv	系统控制单元
core/scr1_tapc.sv	TAP 控制器 （TAPC）
core/scr1_tapc_shift_reg.sv	TAPC 移位寄存器
core/scr1_tapc_synchronizer.sv	TAPC 时钟域交叉同步器
SCR1 排名靠前的集群源文件	
top/scr1_dmem_ahb.sv	数据存储器 AHB 桥
top/scr1_dmem_router.sv	数据存储路由器
top/scr1_dp_memory.sv	带字节使能输入的双端口同步存储器
top/scr1_imem_ahb.sv	指令存储器 AHB 桥
top/scr1_imem_router.sv	指令存储器路由器
top/scr1_mem_axi.sv	存储器 AXI 桥接器
top/scr1_tcm.sv	紧耦合内存 （TCM）
top/scr1_timer.sv	内存映射计时器
top/scr1_top_ahb.sv	SCR1 AHB 顶部
top/scr1_top_axi.sv	SCR1 AXI 顶部
测试台文件	
tb/scr1_memory_tb_ahb.sv	AHB 内存测试台
tb/scr1_memory_tb_axi.sv	AXI 内存测试平台
tb/scr1_top_tb_ahb.sv	SCR1 顶部测试台架 AHB
tb/scr1_top_tb_axi.sv	SCR1 顶部测试台 AXI
tb/scr1_top_tb_runtests.sv	测试台运行测试

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

3. 核心配置

3.1. 核心和设备标识符

下表显示了 SCR1 内核和设备标识符。

表 3：SCR1 内核和设备标识符

标识符名称 描述	
米姆皮德	SCR1 核心实现 ID，以便从相应的 CSR 中读取。该编号唯一标识 SCR1 内核 RTL 的版本。
马尔基德	要从相应的 CSR 中读取的 SCR1 核心架构 ID。该编号标识 SCR1 内核与其他 RISC-V 内核。硬连线到 0x00000008
MVENDORID	要从相应 CSR 读取的 SCR1 核心供应商 ID。出于商业生产目的，请将此字段覆盖为您的 JEDEC 标准制造商 ID 代码。默认值为 0x00000000
TAP_IDCODE	通过 JTAG 从相应的 TAPC 寄存器读取的 IDCODE。出于商业生产目的，请将此字段覆盖为您的 JEDEC 标准制造商 ID 代码 + 零件编号和版本。默认值为 0xDEB11001 （不是 JEDEC）
BUILD_ID	设备内部版本 ID。对于特定设备构建（例如，对于 SCR1-SDK 中的 FPGA 构建），必须在外部 arch_custom.svh 文件中设置该数字。模拟的默认值 = MIMPID

3.2. 推荐配置

下表显示了针对典型使用案例的三种推荐 SCR1 配置。这些配置可以在 scr1_arch_description.svh 文件的 “推荐的核心架构配置” 部分中轻松启用。要选择配置，请从列表中取消注释唯一相关的定义。

表 4：SCR1 推荐配置

选项	SCR1_CFG_RV32EC_MIN	SCR1_CFG_RV32IC_BASE	SCR1_CFG_RV32IMC_MAX
指令集	RV32EC 系列	RV32IC 系列	RV32IMC
管道阶段	2	3	4
GPR 数量	16	32	32
硬件乘数	–	–	+
快速 1 周期乘法器	–	–	+
压缩指令	+	+	+
MTVEC 基址可写位	0	16	26
MTVEC 模式可写	–	+	+
外部 IRQ 线路	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)

选项	SCR1_CFG_RV32EC_MIN	SCR1_CFG_RV32IC_BASE	SCR1_CFG_RV32IMC_MAX
调试子系统	-	+	+
硬件触发器数量	0	2	4
TCM	+	+	+

3.3. 自定义配置的微调选项

SCR1 具有许多用于自定义配置的微调选项，如下表所述。要对这些选项进行自己的设计，您需要编辑 scr1_arch_description.svh 文件：

- 在“推荐的核心架构配置”部分中取消定义所有建议的配置以启用自定义配置，
- 在“自定义核心架构配置”部分中选择所有必要的选项：。要禁用/启用选项 - 注释/取消注释相应的定义，
  - 对于数值参数 - 更改其值。

表 5：SCR1 可配置选项

Name	描述
RISC-V ISA 选项	
SCR1_RVE_EXT	启用 RV32E 基整数指令集，否则将使用 RV32I
SCR1_RVM_EXT	为整数硬件乘法器和除法器启用标准扩展 “M”
SCR1_RVC_EXT	为压缩指令启用标准扩展 “C”
SCR1_MTVEC_BASE_WR_BITS	MTVEC.base 字段中的可写位数
SCR1_MTVEC_MODE_EN	启用可写 MTVEC.mode 字段以允许向量 irq 模式，否则只能使用 direct 模式
核心管道选项（功耗-性能-面积优化）	
SCR1_NO_DEC_STAGE	禁用 IFU 和 IDU 之间的寄存器
SCR1_NO_EXE_STAGE	禁用 IDU 和 EXU 之间的寄存器
SCR1_NEW_PC_REG	启用在 IFU 中为新 PC 值注册
SCR1_FAST_MUL	启用快速单周期乘法，否则乘法需要 32 个周期
SCR1_CLKCTRL_EN	启用全局时钟门控
SCR1_MPRF_RST_EN	启用 MPRF 的重置
SCR1_MCOUNTEN_EN	为计数器控制启用自定义 MCOUNTEN CSR
非核心选项	
SCR1_DBG_EN	启用调试子系统（TAPC、DM、SCU、HDU）
SCR1_TDU_EN	Enable Trigger Debug Unit （hardware breakpoints） （启用触发器调试单元（硬件断点））

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	描述
SCR1_TDU_TRIG_NUM	硬件触发器数量
SCR1_TDU_ICOUNT_EN	在指令计数器上启用硬件触发器
SCR1_IPIC_EN	使能集成可编程中断控制器
SCR1_IPIC_SYNC_EN	为 IRQ 线路启用 2 级输入同步器
SCR1_TCM_EN	Enable Tightly-Coupled Memory，默认大小为 64K

NOTE

对于 synthesis if enable SCR1_CLKCTRL_EN <[>.sv scr1_cg 中的代码应替换为特定于 implementation-memory 的 clock gate。

3.4. 核心集成选项

SCR1 有许多选项可以集成到上层设计中。此选项可以在 scr1_arch_description.svh 文件中的“核心集成选项”部分：

- to disable/enable an options - 注释/取消注释相应的定义，
- 对于数值参数 - 更改其值。

一些选项可以在外部文件 scr1_arch_custom.svh 中定义，该文件未在 SCR1 存储库中提供，但可以在上层项目中使用（例如，打开的 SCR1-SDK 项目和任何其他自定义 FPGA、ASIC 或 SoC 项目）。

表 6：SCR1 集成选项

Name	描述
内存桥旁路选项	
SCR1_IMEM_AHB_IN_BP	在指令存储器 AHB 桥接输入上启用旁路
SCR1_IMEM_AHB_OUT_BP	在指令存储器 AHB 桥接输出上启用旁路
SCR1_DMEM_AHB_IN_BP	在数据存储器 AHB 电桥输入上启用旁路
SCR1_DMEM_AHB_OUT_BP	在数据存储器 AHB 桥接输出上启用旁路
SCR1_IMEM_AXI_REQ_BP	在指令存储器 AXI 桥接请求上启用旁路
SCR1_IMEM_AXI_RESP_BP	在指令存储器上启用旁路 AXI 桥接响应
SCR1_DMEM_AXI_REQ_BP	在数据存储器 AXI 桥接请求上启用旁路
SCR1_DMEM_AXI_RESP_BP	在数据存储器上启用旁路 AXI 桥接响应
地址常量	
SCR1_ARCH_RST_VECTOR	重置向量值（重置后的起始地址）（默认 0x200）
SCR1_ARCH_MTVEC_BASE	MTVEC.base 字段重置值，或硬连线的 MTVEC.base 位的常量值（默认 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)

Name	描述
SCR1_TCM_ADDR_MASK	设置 TCM 蒙版和大小;size in bytes 是 mask 值的 2 的补码（默认 0xFFFF0000）
SCR1_TCM_ADDR_PATTERN	设置 TCM 地址匹配模式（默认 0x00480000）
SCR1_TIMER_ADDR_MASK	设置定时器遮罩（默认 0xFFFFFFE0）
SCR1_TIMER_ADDR_PATTERN	设置计时器地址匹配模式（默认 0x00490000）
目标平台（启用特定于目标的构建）	
SCR1_TRGT_FPGA_INTEL	目标平台是英特尔 FPGA
SCR1_TRGT_FPGA_INTEL_MAX10	目标平台是英特尔 MAX 10 FPGA（用于 SCR1-SDK 项目）
SCR1_TRGT_FPGA_INTEL_ARRIAV	目标平台是英特尔 Arria V FPGA（用于 SCR1-SDK 项目）
SCR1_TRGT_FPGA_XILINX	目标平台是 Xilinx FPGA（用于 SCR1-SDK 项目）
SCR1_TRGT_ASIC	目标平台是 ASIC

3.5. 模拟选项

以下参数仅用于仿真，以启用仿真代码和设置测试平台。这些选项可以在 scr1_arch_description.svh 文件的“SIMULATION OPTIONS” 部分中更改：

- to disable/enable an options - 注释/取消注释相应的定义，
- 对于数值参数 - 更改其值。

表 7：SCR1 仿真选项

Name	描述
模拟选项	
SCR1_TRGT_SIMULATION	启用模拟代码（由根 makefile 自动定义）（请参阅模拟代码）
SCR1_TRACE_LOG_EN	启用 tracelog（请参阅 Tracelog）
Testbench 中使用的地址（请参阅 Testbench 描述）	
SCR1_SIM_EXIT_ADDR	写入此地址以退出模拟（默认为 0x000000F8）
SCR1_SIM_PRINT_ADDR	写入此地址以在 console 中打印元件（默认 0xF0000000）
SCR1_SIM_EXT_IRQ_ADDR	写入此地址以生成外部中断（默认 0xF0000100）
SCR1_SIM_SOFT_IRQ_ADDR	写入此地址以生成软件中断（默认 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

4. 模拟环境

该项目包含测试平台、测试源和脚本，用于快速启动 SCR1 仿真。在开始模拟之前，请确保您已：

- 已安装 RISC-V GCC 工具链，
- 安装其中一个受支持的模拟器，
- 使用测试源初始化子模块。

4.1. 要求

4.1.1.作系统

大多数流行的类 Linux 作系统都支持 GCC 工具链和 make-scripts。

要从 Windows 运行，您可以使用额外的兼容层，例如 WSL 或 Cygwin。

4.1.2. RISC-V GCC 工具链

编译软件需要 RISC-V GCC 工具链。您可以使用预构建的二进制文件或从头开始构建工具链。

4.1.2.1. 使用预构建的二进制工具

支持所有 SCR1 架构配置的预构建 RISC-V GCC 工具链可从 http://syntacore.com/page/products/sw-tools 下载。

1. 下载适用于您的平台的存档。
2. 将存档文件解压到首选目录 <GCC_INSTALL_PATH>。
3. 将 <GCC_INSTALL_PATH>/bin 文件夹添加到 $PATH 环境变量中：export PATH=<GCC_INSTALL_PATH>/bin：$PATH

4.1.2.2. 从源构建工具

您可以从存储在官方仓库 https://github.com/ riscv/riscv-gnu-toolchain 的源代码构建 RISC-V GCC 工具链
有关如何准备和构建工具链的说明，请访问 riscv-gnu-toolchain/blob/master/README.md
https://github.com/riscv/
我们建议使用 multilib 编译器。请注意，默认情况下，编译器中不包含 RV32IC、RV32E、RV32EM、RV32EMC、RV32EC 架构配置。如果您打算使用它们，则需要在构建之前自己包含适当的库。
构建后，请务必将 <GCC_INSTALL_PATH>/bin 文件夹添加到 $PATH 环境中

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>`,

变量

4.1.3. HDL 模拟器

当前支持的模拟器：

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

请注意，RTL 模拟器可执行文件应位于 $PATH 变量中。

4.1.4. 测试准备

模拟包包括以下测试：

- hello - “Hello” 示例程序
- isr_sample - “Interrupt Service Routine” 示例程序
- riscv_isa - RISC-V ISA 测试（子模块）
- riscv_compliance - RISC-V 一致性测试（子模块）
- dhrystone21 - Dhrystone 2.1 基准
- coremark - EEMBC 的 CoreMark® 基准测试（子模块）

克隆主 SCR1 存储库后，执行以下命令：

```
git submodule update --init --递归
```

此命令将使用测试源初始化子模块。

4.2. 运行模拟

要构建 RTL，请从必须调用 Makefile 的 repo 根文件夹编译并运行测试。默认情况下，您可以简单地调用 Makefile，不带任何参数：

```
make
```

在这种情况下，模拟将在 Verilator 上运行，参数如下：CFG=MAX BUS=AHB TRACE=0 TARGETS= “hello isr_sample riscv_isa riscv_compliance dhrystone21 coremark”。

Makefile 支持：

- 模拟器的选择 - 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)

- 架构设置 - CFG、BUS、ARCH、VECT_IRQ、IPIC、TCM、
- 要运行的测试子集 - TARGETS
- 启用 tracelog - TRACE
- 以及要传递给模拟器的任何其他选项 - SIM_BUILD_OPTS。

例：

```
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”
```

可以在 ./Makefile 中配置生成和运行参数。

所有测试完成后，结果可以在 build/<SIM_CFG>/test_results.txt 中找到。

重要 为确保正确重建，请在仿真运行之间调用 make clean。

4.2.1. 模拟器选择

您可以指定一个受支持的模拟器 run_<SIMULATOR> = <run_vcs、run_modelsim、run_ncsim、run_verilator run_verilator_wf>：

```
制作 run_modelsim
```

模拟器运行：

- run_verilator - Verilator （默认）
- run_verilator_wf - 带波形生成的 Verilator
- run_modelsim - Mentor Graphics 或 Intel 的 ModelSim
- run_vcs - Synopsys VCS
- run_ncsim - Cadence NCSim

对于 run_verilator_wf 选项，将为执行的所有测试生成波形，并将其保存在 ./build/<SIM_CFG>/simx.vcd 中。该文件可以通过某些波形查看器打开，例如 GTKWave。

4.2.2. 架构配置

您可以指定配置 CFG = <MAX、BASE、MIN、CUSTOM>和外部接口 BUS = <AHB, AXI>：

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

配置扩展如下：

- MAX - 设置预定义的配置 SCR1_CFG_RV32IMC_MAX（默认）

- **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
```

- BASE - 设置预定义的配置 SCR1_CFG_RV32IC_BASE
- MIN - 设置预定义的配置 SCR1_CFG_RV32EC_MIN
- CUSTOM - 可用于任何其他自定义配置

对于所有预定义配置，其他架构参数会自动设置为确定性状态，用于编译测试和 SCR1 RTL。

对于 CUSTOM 配置，您可以指定其他参数：

- ARCH = <IMC、IC、IM、I、EMC、EM、EC、E> - RISC-V 指令集架构。该参数定义了用于编译测试的 RISC-V 指令集架构（由 RISC-V 工具链自动使用）：RV32I 或 RV32E 基础 + 可选的标准扩展 M 和 C。RTL 选项 SCR1_RVE_EXT、SCR1_RVM_EXT 和 SCR1_RVC_EXT 必须相应地定义。

- VECT_IRQ = <0， 1> - 用于处理中断的矢量模式，否则使用直接模式。参数 VECT_IRQ 的定义在测试 isr_sample 中用于显示各种中断调用和处理场景。必须为 vectored 模式定义 RTL 选项 SCR1_MTVEC_MODE_EN。

- IPIC = <0， 1> - 使用集成可编程中断控制器。参数 IPIC 的定义在测试 isr_sample 中用于显示各种中断调用和处理场景。RTL 选项 SCR1_IPIC_EN 必须相应地定义。

- TCM = <0， 1> - 使用紧密耦合内存。将 TCM 选项设置为 1 可定义要从 Tight Coupled Memory 而不是外部 testbench 内存执行的一些测试。RTL 选项 SCR1_TCM_EN 必须相应地定义

重要	CUSTOM 配置的附加参数集不会启用 SCR1 RTL 参数。请不要忘记在文件 ./src/includes/scr1_arch_description.svh 中手动设置相应的参数。

NOTE	其他参数不能用于预定义配置，因为它们已经硬编码。

例：

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

4.3. 目标

您可以指定要在模拟中运行的测试子集：

- 目标 = <hello、isr_sample、riscv_isa、riscv_compliance、dhrystone21、coremark>

要从列表中仅选择一个目标，请指定其名称，例如：

```
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.

要选择多个目标，请用引号将它们列出，并用空格分隔，例如：

```
make TARGETS= “dhrystone21 coremark”
```

某些测试取决于所选架构，因此不能用于所有核心配置（这些配置会自动跳过）。

要从集合中选择单个测试，您需要：

- 对于 riscv_isa 集合，请转到 ./sim/tests/riscv_isa/rv32_tests.inc 并在 rv32_isa_tests 列出所需的测试。
- 对于 riscv_compliance 集合，请转到 ./sim/tests/riscv_compliance/Makefile 并在 compliance_set 中列出所需的测试 （指定每个测试的完整路径）。

4.4. 模拟代码

您可以添加有关仿真过程的有用信息：断言、tracelog 和 instruction statistics。必须在 ./src/includes/scr1_arch_description.svh 部分的“SIMULATION OPTIONS”中定义 SCR1_TRGT_SIMULATION 参数才能启用所有模拟代码。当您运行 make-script 时，此参数会自动启用。

4.4.1. 跟踪日志

在模拟过程中，可以将以下信息写入特殊文件
tracelog_core_N.log build 目录中：

- RTL_ID 值
- 核心重置事件及其发生时间
- MPRF 和 CSR 寄存器按以下格式更新信息：时间 |活动 |Curr_PC |Instr |Next_PC |注册 |价值

使用以下事件的缩写：

- N - 无事件（常规寄存器值更新或无更新的循环）
- E - 异常
- I - 中断
- W - 唤醒

必须在 src/includes/scr1_arch_description.svh 中定义参数 SCR1_TRACE_LOG_EN 和 SCR1_TRGT_SIMULATION 才能启用 tracelog。使用 make-script 时，您可以传递参数 TRACE=1 以自动为所有选定的测试启用跟踪日志生成。

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.

4.5. 测试平台描述

SCR1 测试平台由顶级 模 块 和 外部 存储器 组成。根据所使用的存储器接口，有两种测试平台配置可供选择：AXI 或 AHB。
可以通过指定 make 命令的 BUS=<AHB， AXI> 选项来选择所需的配置

（有关完整命令，请参阅 [运行模拟]）。默认值为 BUS=AHB。

下表提供了两种配置的文件列表。

表 8：SCR1 测试台文件

Path	描述
AXI 测试平台	
tb/scr1_memory_tb_axi.sv	AXI 内存测试平台
tb/scr1_top_tb_axi.sv	SCR1 顶部测试台 AXI
AHB 测试台	
tb/scr1_memory_tb_ahb.sv	AHB 内存测试台
tb/scr1_top_tb_ahb.sv	SCR1 顶部测试台架 AHB

两个 testbench 存储器的大小均为 1024 kB。

NOTE

如果启用了 TCM，则其内存地址范围将从外部内存地址范围中剪切。

Testbench 存储器提供了通过将数据写入特定地址在仿真控制台中生成中断和打印字符的机制。同时加载具有 Simulation Exit address 值的 Program Counter 将终止仿真。此类地址的定义位于 scr1_arch_description.svh 的 “SIMULATION OPTIONS” 部分。默认地址映射如表 1 所示。

表 9：SCR1 中断和仿真控制默认内存映射

地址	描述
0xF0000100	外部 IRQ
0xF0000200	软件 IRQ
0xF0000000	打印字符
0x000000F8	模拟退出

外部 1 引脚 IRQ 和 IPIC IRQ 线路共享同一地址。确保您使用的宽度正确，具体取决于是否启用了 IPIC。外部 1 引脚 IRQ （如果禁用了 IPIC）和软 IRQ 值应放在第 0 位。IRQ 行值（如果启用了 IPIC）使用 16 个最低有效位的写入数据。

NOTE

如果需要使用 default 以外的 addresses map，请确保 RTL 和 test program 使用相同的 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

5. SDK 信息

https://github.com/syntacore/scr1-sdk 上提供了基于 FPGA 的开源 SDK。

存储库包含：

- 为多个标准 FPGA 板提供预构建映像和开放式设计：◦ Digilent Arty （Xilinx）

    ◦ Digilent Nexys 4 DDR （赛灵思）

    ◦ Arria V GX 起动器 （Intel）

    ◦ Terasic DE10-Lite （英特尔）
- 软件包： ◦ Bootloader

    ◦ Zephyr 实时作系统

    ◦ 测试和软件样本
- SDK 和工具的用户指南

6. Support

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

6. 支持

有关 SCR1 内核的更多信息，请写信给 scr1@syntacore.com。