Published by IOP Publishing for Sissa Medialab

Received: *November 3, 2024*
Accepted: *January 23, 2025*
Published: *February 11, 2025*

Topical Workshop on Electronics for Particle Physics
University of Glasgow, Scotland, U.K.
30 September–4 October 2024

SOCRATES: a radiation-tolerant SoC generator framework

Marco Andorno,* Alessandro Caratelli, Davide Ceresa, Benoît Denkinger, Kostas Kloukinas, Anvesh Nookala and Risto Pejašinović

CERN,
Geneva, Switzerland

E-mail: marco.andorno@cern.ch

Abstract: As front-end ASIC complexity in HEP experiments grows, there is a shift towards more modular, programmable, and cost-effective designs. This work introduces the SOCRATES platform, a radiation-tolerant SoC generator toolset, based on SoCMake, a hardware/software build system that automates SoC assembly and verification. Utilizing existing IP blocks, SoCMake generates the SoC hardware and the software framework to run application code. The platform includes radiation-tolerant IPs and fault-tolerant extensions supporting redundancy and error correction. A prototype ASIC based on the RISC-V Ibex processor, generated using SOCRATES in a 28nm CMOS process, will validate the toolset through SEE and TID testing.

Keywords: Digital electronic circuits; Radiation-hard electronics; VLSI circuits

*Corresponding author.

https://doi.org/10.1088/1748-0221/20/02/C02017

Contents

1 Introduction

As the complexity of High-Energy Physics (HEP) experiments continues to increase, the electronics will need to scale accordingly and provide the required performance. Front-end ASICs in particular will need to deal with more data and integrate more functionalities than ever before, while retaining exceptional reliability under high-radiation environments [1]. The use of ultra deep submicron technology nodes will be necessary to meet these requirements. This, in turn, will drive up significantly the ASIC development costs. One way of mitigating this is a shift towards programmable System-on-Chips (SoC) ASICs. By introducing software programmability, the same front-end ASIC can be re-targeted for different applications. This results in a smaller number of ASICs needing to be designed. Moreover a comprehensive SoC ASIC development platform will enable faster design and verification turnaround times, resulting in more cost-effective development.

This work presents SOCRATES, a toolset to generate radiation-tolerant SoCs, developed within the context of the CERN EP R&D WP5 framework, building on the work presented in [2]. The requirement for an SoC generator arises from the principle of having modular designs, which can be quickly composed starting from available building blocks, and the need to have a diverse range of SoC architectures, to target different applications with different performance, power and area trade-offs. SoC generators and hardware build systems are not new [3–5], but they either support just a single type of processor, they do not offer a fully integrated set of tools, or, most importantly, they do not have any support for fault tolerance, which is key for HEP applications. SOCRATES aims at being a toolset generic enough to be used in a wide range of applications, but at the same time easily extensible for domain-specific requirements.

2025 JINST 20 C02017

2 The SOCRATES toolkit

SOCRATES (SOC RAdiation Tolerant EcoSystem) is a flexible all-in-one solution for the bring-up of custom SoCs. Its HW/SW build system, SoCMake, see section 3, provides tools to build a modular system from the Verilog description up to the software stack, where the processing core(s), the memory system, the buses, and the peripherals can be selected freely, composed, and connected together automatically, as shown in figure 1.

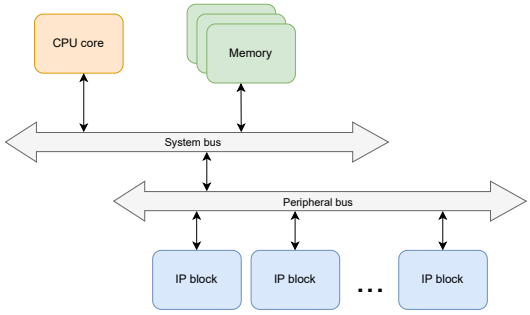

Figure 1. Basic SoC architecture that SOCRATES can generate.

SOCRATES already supports a number of RISC-V CPU cores, namely lowRISC's Ibex [6], CHIPS Alliance's VeeR EL2 [7], Syntacore's SCR1 [8], and Yosys' PicoRV32 [9]. These RISC-V cores were selected during development of the ecosystem due to their availability as open-source solutions and because RISC-V, being an open standard extensible Instruction-Set Architecture (ISA) [10], best suits the needs of this R&D. However, limited amount of effort will be required to support additional cores and potentially other ISAs. A suitable set of fast interconnect buses is also supported for the connection of CPU(s) and memories, such as AMBA AXI4 and OpenHW OpenBus Interface (OBI) [11].

2.1 APB-RT

The main goal of SOCRATES is to provide tools for HEP microelectronics designers to conveniently build and integrate SoCs into their specific applications. For this reason, to engage the community and promote collaborative work, a common peripheral bus for all IP blocks within the platform is proposed. By making sure that all IPs conform to a standardized interface, modularity and design reuse become much easier to achieve.

The proposed standardized interface is a custom version of the AMBA APB5 protocol [12], modified for radiation tolerance, called APB-RT. The standard APB control signals are triplicated, while the 32-bit data and address buses are encoded with a Hamming(13,8) encoding by byte, to permit easy byte-access operations. The parity bits of the encoding for each byte are used for Single Error Correction, Double Error Detection (SECDED): this scheme can correct up to four errors in a word, if they occur in different bytes, and can otherwise propagate an error signal to the CPU to engage a software error recovery routine. Compared to triplication, this encoding reduces the number of individual wires by 66%, significantly saving routing resources and power and easing implementation [13].

3 SoCMake

SoCMake is a CMake-based build system developed for the HW/SW co-design of SoCs. It manages the dependencies between the different IP blocks, invokes the correct toolchain for code cross-compilation, and can build outputs for different hardware targets. More importantly, SoCMake can automatically generate both hardware and software components, starting from a single common architectural description of the system written in SystemRDL, see section 3.1. Figure 2 shows a diagram of the inputs required by SoCMake and the different targets it can build.

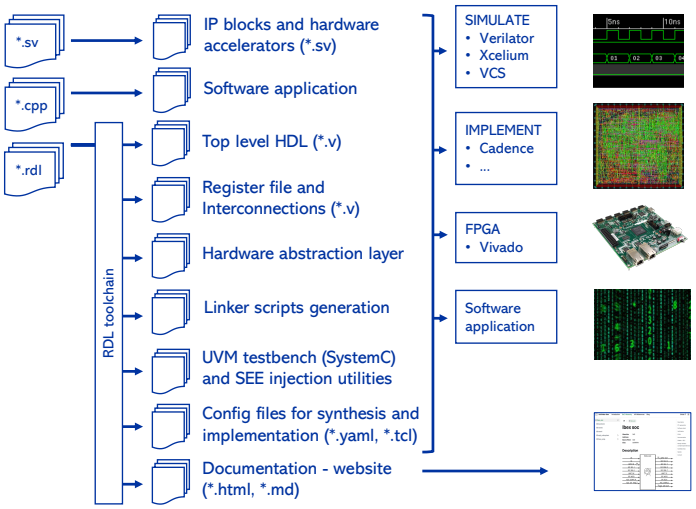

Figure 2. SoCMake build system structure overview.

At the core of SoCMake is the hardware IP (HWIP), which is defined as a CMake interface library, specified in the VLNV (Vendor, Library, Name, Version) format. A HWIP can be anything from peripheral IP blocks, to interconnects, to full SoCs. Each HWIP has different sources, that can be RTL files, SystemRDL descriptions, C/C++ files, Markdown documentation, etc. Then, SoCMake provides a number of CMake functions to build targets for different operations on HWIPs, such as generating or modifying sources, adding dependencies between them, compiling code, or invoking EDA tools.

These functionalities are the key for quick SoC prototyping, where different architectures can be easily generated by composition of different blocks, from simple microcontrollers to advanced SoCs with custom hardware accelerators. SoCMake is available open-source on GitHub[1] so that any welcome contribution from the community can help improve it and expand its use cases.

3.1 SystemRDL and the PeakRDL toolchain

To infer connections between components and construct the full system description, SoCMake relies on a SystemRDL description. SystemRDL is an Accellera standard originally intended to be used as a register description language, however, in SoCMake, some custom properties were added to effectively use it as a top-level system description as well. Each HWIP includes a SystemRDL description that not only lists its configuration registers and fields but also details the interfaces it exposes, any sub-IPs, and how they are interconnected.

[1] https://github.com/HEP-SoC/SoCMake/.

In order to process the SystemRDL description files, the PeakRDL[2] suite of tools is used within SoCMake. PeakRDL combines multiple SystemRDL files for different HWIP and provides a number of plugins to compile and generate outputs from them, notably: *PeakRDL-regblock* generates synthesizable control and status register (CSR) blocks in SystemVerilog; *PeakRDL-html* generates an HTML documentation of the register map; *PeakRDL-uvm* generates a UVM RAL register model.

Several custom PeakRDL plugins were developed and are supported by SoCMake: *PeakRDL-socgen* generates the top-level Verilog description of the SoC, by automatically instantiating sub-blocks and inferring the proper interconnect between them; *PeakRDL-halcpp* generates the C++ Hardware Abstraction Layer (HAL) to streamline peripheral code writing without adding to the code size; *PeakRDL-opentitan* helps converting the Hjson description of OpenTitan[3] IPs to SystemRDL, facilitating IP reuse from other open-source hardware projects; *PeakRDL-ipblocksvg* and *PeakRDL-docusaurus* can automatically generate HWIP documentation as well.

3.2 Add-ons for radiation tolerance

SoCMake can be easily extended by adding custom CMake functions to expand the core set of functionalities. Within SOCRATES, a number of add-ons are present to implement radiation tolerance in the toolkit.

In terms of hardware composition, among the supported interconnects, there are now fault-tolerant bus protocols, such as the APB-RT described in section 2.1. A custom target can be defined to automatically apply Triple Modular Redundancy (TMR) to the RTL sources of any HWIP by invoking the TMRG tool [14].

In order to verify that TMR was applied correctly to the logic, SoCMake can also launch the TMR formal verification methodology presented in [15]. Finally, a complete fault injection framework is also supported within the SOCRATES ecosystem.

4 TriglaV: a demonstrator chip for SOCRATES

In order to demonstrate that the hardware generated by SoCMake works in silicon, a radiation-tolerant microcontroller-like SoC called TriglaV has been designed using the SOCRATES tools, with a focus on validating the toolkit and demonstrating the potential of integrating programmable logic in front-end ASICs. The goals for this demonstrator chip will also involve testing its radiation performance and gaining experience in SoC design for high-reliability applications.

The SoC is fully generated from the SOCRATES platform with a suitable set of peripherals for a potential application as an embedded controller. It is designed for radiation tolerance up to approximately CMS Inner Tracker levels, and incorporates testability features to trace faults under irradiation. The chip is implemented on a commercial 28 nm bulk CMOS technology (figure 3), with a prototype tapeout on multi-project wafer (MPW). The target operating frequency is 250 MHz and the area is approximately 1×2 mm. An FPGA emulation has been set up to increase the confidence on the generated hardware, while extensive simulations of the top-level and random verification of the peripherals have taken place.

[2] https://peakrdl.readthedocs.io/.
[3] https://opentitan.org/.

2025 JINST 20 C02017

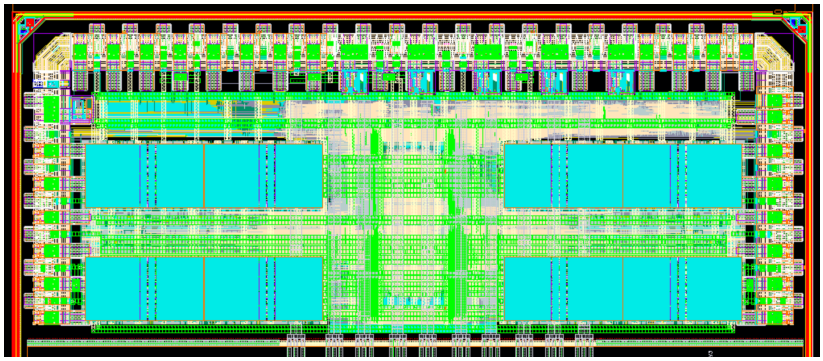

Figure 3. TriglaV layout.

4.1 Architecture

TriglaV uses the Ibex core, a 32-bit, RV32IMC, 2-stage pipeline, RISC-V core that provides a good trade-off between performance and area. The original RTL code of the processor has been fully triplicated and majority voters inserted at the flip-flop level to correct single-event upsets (SEU) within one clock cycle, resulting in a 4x area increase over the original size. The complete coverage of the circuitry by the TMR scheme has been verified with extensive formal verification and fault injection campaigns.

The memory system consists of a hardwired bootloader to load the code to the memory through a Universal Asynchronous Receiver-Transmitter (UART) interface and two dedicated Memory Scrubbing and Protection Units (MSPU) for instruction and data. The MSPUs feature dual-port commercial SRAM blocks for a total of 32 kB each with physical bit interleaving and configurable periodic scrubbing. Data ECC encoded by byte with a Hamming (13,8) code. Memories and CPU are connected by a triplicated Open Bus Interconnect (OBI) bus (OBI-TMR) multi-master crossbar.

Peripherals include the aforementioned UART, a RISC-V privileged architecture compliant machine timer, 8 General Purpose Input Output (GPIO) channels and a Platform-Level Interrupt Controller (PLIC). These have mostly been taken from the open-source hardware community, such as the PULP, lowRISC, and OpenHW group projects, and modified to be compatible with SoCMake and TMRG. All peripherals are connected via the APB-RT interface described in section 2.1.

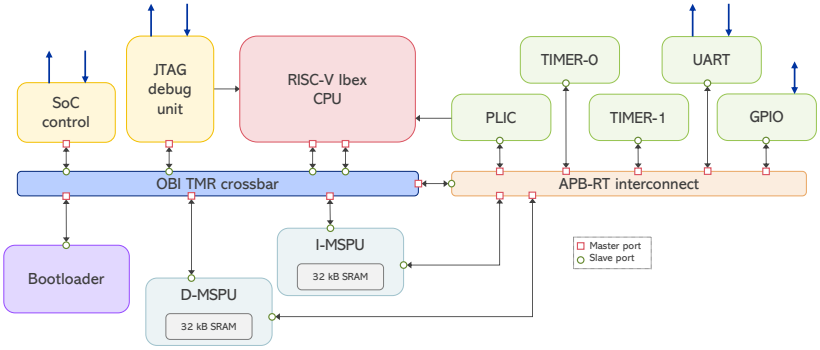

Figure 4. TriglaV architecture.

2025 JINST 20 C02017

4.2 Fault tolerance and debug features

A debug unit featuring a JTAG port with access to the whole address space through the OBI-TMR interconnect is going to be used for general on-chip software debugging.

Additionally, a full-TMR SoC control unit with a dedicated I2C interface is also present. This provides a redundant boot mechanism, allowing the I2C to function as an additional master on the OBI-TMR bus. The same interface is also used to read out SEU counters throughout the chip, in order to gather SEU statistics under irradiation testing.

Finally, some chip outputs are dedicated to special signals coming from the CPU and memory blocks. During irradiation, these signals can be compared with a reference simulation output running the same code, allowing for the identification of when and where errors occur, albeit at a coarse granularity.

5 Summary and future outlook

The present work introduces SOCRATES, a set of tools to build radiation-tolerant SoCs. At its core are the SoCMake HW/SW build system, which brings together the SoC and generates hardware and software components, and a set of fault tolerance add-ons. The ecosystem provides a collection of pre-qualified IP blocks, all conforming to a common bus protocol, namely APB-RT. This library of IPs is actively developed, maintained, and extended by the SOCRATES core team, but a broader active collaboration is sought to further enlarge it. The aim is to foster collaboration within the HEP community and beyond to expand the proposed ecosystem's applications.

To demonstrate on silicon the SOCRATES tools, a prototype chip named TriglaV has been developed, with a focus on being a minimal radiation-tolerant microcontroller-like SoC with suitable observability features for tracing errors under irradiation testing campaigns.

The future plans for this R&D include expanding and refining the functionality of the SOCRATES tools to enhance user-friendliness and open-sourcing additional components of the platform. Moreover, the plans involve evaluating alternative solutions to TMR for designing fault tolerant SoC architectures and demonstrate the feasibility of integrating similar systems in larger front-end chips, such as pixel-readout ASICs, or transitioning towards high performance architectures with custom hardware accelerators.

References

[1] ECFA Detectors R&D Roadmap Process Group, *The 2021 ECFA Detector Research And Development Roadmap*, CERN-ESU-017 (2021).

[2] M. Andorno et al., *Radiation-Tolerant SoC and Application-Specific Processors for On-Detector Programmability and Data Processing in Future High-Energy Physics Experiments*, in the proceedings of the *12th International Conference on Modern Circuits and Systems Technologies*, Athens, Greece (2023), p. 1–5 [DOI:10.1109/MOCAST57943.2023.10176589].

[3] K. Asanović et al., *The rocket chip generator*, EECS Department, University of California, Berkeley (2016) [UCB/EECS-2016-174].

[4] Antmicro, *SoC Generator*, https://github.com/antmicro/soc-generator.

[5] O. Kindgren, *FuseSoC*, https://fusesoc.readthedocs.io.

2025 JINST 20 C02017

[6] P. Davide Schiavone et al., *Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications*, in the proceedings of the *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Thessaloniki, Greece (2017), p. 1–8 [DOI:10.1109/patmos.2017.8106976].

[7] CHIPS Alliance, *VeeR EL2 RISC-V Core*, https://github.com/chipsalliance/Cores-VeeR-EL2.

[8] Syntacore, *SCR1 RISC-V Core*, https://github.com/syntacore/scr1.

[9] C. Wolf, *PicoRV32 — A Size-Optimized RISC-V CPU*, https://github.com/YosysHQ/picorv32.

[10] A. Waterman, Y. Lee, D. Patterson and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0*, EECS Department, University of California, Berkeley (2014) [UCB/EECS-2014-54].

[11] OpenHW Group, *OpenBus Interface v1.6.0*, https://github.com/openhwgroup/obi/blob/main/OBI-v1.6.0.pdf.

[12] ARM Ltd, *AMBA APB Specification (Issue D)*, https://documentation-service.arm.com/static/60d5b505677cf7536a55c245?token= (2021).

[13] M. Andorno et al., *Rad-hard RISC-V SoC and ASIP ecosystems studies for high-energy physics applications*, 2023 *JINST* **18** C01018.

[14] S. Kulis, *Single Event Effects mitigation with TMRG tool*, 2017 *JINST* **12** C01082.

[15] A. Pulli and M. Lupi, *A simulation methodology for verification of transient fault tolerance of ASICs designed for high-energy physics experiments*, 2023 *JINST* **18** C01038.

2025 JINST 20 C02017