Automate Fault-Tolerant SoC Generation with the SOCRATES Platform

Marco Andorno[1][*], Alessandro Caratelli[1], Benoît Denkinger[1], Kostas Kloukinas[1], Anvesh Nookala[1]

[1]CERN - 1211 Geneva 23 - Switzerland

Abstract

SOCRATES is a SoC generator framework, specifically targeted at fault-tolerant architectures, developed for high-energy physics experiments, but extensible wherever high reliability is required. It allows to automatically build full SoCs starting from a collection of RISC-V CPU cores, IP blocks and interconnects. To do that, a custom build system, called SoCMake, takes care of managing dependencies and invoking the tools to generate both hardware and software components of the system. A silicon prototype has been designed and taped out to validate the toolkit and test the performance of the resulting architecture.

Introduction

As the complexity of system-on-chips (SoCs) continues to increase, ASIC design is becoming more and more costly and time-consuming, particularly when considering ultra-deep submicron technology nodes. To tackle these challenges, heavily relying on IP block re-use and exploiting the modularity of integrated systems can greatly help reduce design and verification turnaround time, offering quicker system prototyping, a smaller number of reusable ASICs and eventually a more cost-effective development overall.

This is especially true in the high-energy physics (HEP) domain, where experiments will continue to push the limits of data acquisition and processing. Additionally, the electronics used for HEP experiments need to ensure reliable operation even under extremely high radiation levels. For this reason, ICs designed for HEP experiments have historically always been relying on application-specific optimizations and custom design techniques for fault-tolerance, thus making it difficult to integrate and re-use third-party IPs.

To address these challenges, this work introduces SOCRATES (**S**ystem **o**n **C**hip **RA**diation-**T**olerant **E**co**S**ystem), a flexible radiation-tolerant SoC generator toolkit, that automates the design and verification of fault-tolerant RISC-V-based SoCs. Centered around a hardware/software co-design build system called SoCMake [1], SOCRATES enables the rapid assembly of modular SoC architectures, from the hardware description to the firmware stack, integrating redundancy and error correction for the high reliability required in HEP ASICs. By offering a configurable, reusable, and scalable approach, SOCRATES significantly reduces development effort and costs, making it a suitable solution for a wider range of fault-tolerant applications even beyond HEP.

[*]Corresponding author: `marco.andorno@cern.ch`

SOCRATES components

The SOCRATES toolkit is comprised of a collection of pre-verified IP blocks and build tools for SoCs generation. The processing core(s), the memory sub-system, the buses, and the peripherals can be selected freely, composed, and assembled automatically. The platform already supports multiple RISC-V cores, including lowRISC's Ibex, CHIPS Alliance's VeeR EL2, Syntacore's SCR1, and Yosys' PicoRV32.

Each peripheral IP block adheres to a common 32-bit interconnect protocol called APB-RT, a modified version of the AMBA APB5 standard, where the data and address buses use Hamming encoding for ECC and the control lines are triplicated for triple modular redundancy (TMR) to ensure fault-tolerance. This common bus makes it easy to add custom peripheral ensuring compatibility.

The build system requires as input a single common description of the top-level system and the IP blocks written in SystemRDL language, to generate hardware and software components of the final SoC.

The SoCMake build system

At the core of the SOCRATES platform, is SoCMake, an open-source CMake-based build system that manages the dependencies between hardware IP blocks and software components, invokes the toolchain for code cross-compilation for RISC-V and builds the outputs for different hardware targets, such as simulation, FPGA emulation and ASIC implementation (fig. 1).

SoCMake is an API layer on top of CMake which enables the definition of hardware design build flows in a similar fashion to how it is commonly done in

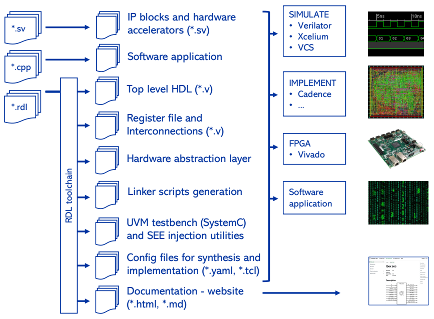
*.sv
*.cpp
*.rdl
RDL toolchain
IP blocks and hardware accelerators (*.sv)
Software application
Top level HDL (*.v)
Register file and Interconnections (*.v)
Hardware abstraction layer
Linker scripts generation
UVM testbench (SystemC) and SEE injection utilities
Config files for synthesis and implementation (*.yaml, *.tcl)
Documentation - website (*.html, *.md)
SIMULATE
• Verilator
• Xcelium
• VCS
IMPLEMENT
• Cadence
• ...
FPGA
• Vivado
Software application

Figure 1: *SoCMake structure*

software projects. SoCMake introduces the Hardware IP (HWIP) library, defined as a CMake interface library, which can be anything from peripheral IP blocks, CPUs, interconnect protocols definitions, up to full SoCs. Each HWIP incorporates various sources, including RTL files, SystemRDL descriptions, C/C++ code, and Markdown documentation. SoCMake then offers a set of CMake functions to define build targets for HWIP operations, such as source generation or modification, dependency management, code compilation, and EDA tool invocation.

To infer the connections between different HWIPs and build the top-level SoC, SoCMake uses a description in SystemRDL, an Accellera standard register description language, that, in this context, is used also as an architectural description. The SystemRDL files of all the HWIPs are compiled together by an open-source SystemRDL Compiler[1] and then used to generate different components of the SoC with plugins, some available from the PeakRDL[2] suite, some custom developed. As an example (non exhaustive list): *PeakRDL-regblock* generates synthesizable control and status register (CSR) blocks in SystemVerilog; *PeakRDL-uvm* generates a UVM RAL register model; *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.

SoCMake is easily expandable to support custom tools for, for instance, manipulate the RTL code introducing fault-tolerance techniques as Triple Modular Redundancy (TMR). In general, SoCMake aims to be a lightweight build system easily extendable with any additional functions or tools that a specific application may require. SoCMake is available open-source and any contribution is welcome.

[1] `https://systemrdl-compiler.readthedocs.io`
[2] `https://peakrdl.readthedocs.io/en/latest`

Silicon prototype

To validate SOCRATES and its toolset, a radiation-tolerant microcontroller-like SoC prototype, called TriglaV, was designed and prototyped in a commercial 28 nm CMOS process, (fig. 2). TriglaV features a triplicated Ibex RISC-V core, protected with TMR at the flip-flop level, ensuring Single-Event Upset (SEU) correction within a single clock cycle. The memory system includes dual-port SRAMs with periodic scrubbing and hamming-based error-correcting code techniques (ECC) for high reliability. The processor core and memory subsystem are interconnected via an OBI-TMR bus, ensuring end-to-end fault tolerance.

TriglaV integrates a few essential peripherals, including a JTAG debug unit, a Platform-Level Interrupt Controller (PLIC), machine timers, GPIOs, and a UART module. These peripherals, sourced from open hardware projects like PULP and OpenTitan, were modified to ensure compatibility with the SOCRATES ecosystem. Additionally, in order to be able to test the performance of the system also under radiation testing, TriglaV features redundant booting mechanisms (UART, I2C, and JTAG), error counters, and fast debug outputs providing a hash of critical CPU signals.

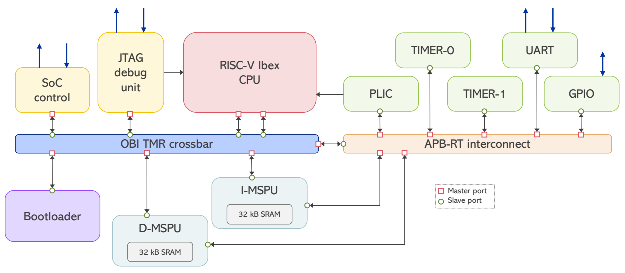

Figure 2: *TriglaV architecture*

To ensure robustness and reliability, the TriglaV prototype chip has undergone extensive pre-silicon validation using formal verification techniques, simulation campaigns with fault injection, and FPGA-based emulation. The next phase involves irradiation experiments with heavy ions and laser beams to assess the fault tolerance capability of the design and to correlate it with the simulation model, providing valuable insights into its performance under extreme conditions. The results from these tests will further refine the SOCRATES framework and guide future developments in radiation-hard SoC architectures.

References

[1] Risto Pejašinović et al. *Hardware and software build flow with SoCMake.* 2025. arXiv: `2502.02065 [cs.AR]`. URL: `https://arxiv.org/abs/2502.02065`.