SafeTI Traffic Injector Enhancement for Effective Interference Testing in Critical Real-Time Systems

Francisco Fuentes[1,2], Raimon Casanova[2], Sergi Alcaide[1] and Jaume Abella[1]

[1]Barcelona Supercomputing Center (BSC), Barcelona, Spain
[2]Microelectronic and Electronic Systems Department, Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain

Abstract

Safety-critical domains, such as automotive, space, and robotics, are adopting increasingly powerful multicores with abundant hardware shared resources for higher performance and efficiency. However, mutual interference due to parallel operation within the SoC must be properly validated. Recently, the SafeTI traffic injector has been released and integrated in a homogeneous RISC-V multicore for testing, otherwise untestable casuistic for software-only solutions. This paper introduces some enhancements performed on the SafeTI, which include internal pipelining for higher-rate traffic injection, and its tailoring to multiple interfaces, as well as its integration in a more powerful heterogeneous RISC-V multicore based on Gaisler's technology for the space domain.

Introduction

Increasing performance demands in safety-critical real-time systems impose the adoption of Multi-Processor Systems-on-Chip (MPSoCs). MPSoCs include multiple cores and accelerators that run several tasks in parallel, sharing hardware resources for efficiency reasons (e.g., on-chip caches, memory controllers, I/O interfaces). Such sharing brings timing interference across cores, accelerators, and I/O interfaces, since several devices may contend for access to a specific device unable to serve requests from different sources simultaneously. Hence, requests are serialized, causing execution time delays on the affected tasks.

The development process of safety-critical systems is guided by domain-specific safety standards and guidelines (e.g., ISO26262 for automotive) that impose safety requirements to the system, which often include real-time requirements (e.g., braking the car within a specific timeframe since the braking pedal is activated). The architectural design of the system is devised to meet those safety requirements by construction, but the safety development process also imposes testing on the system to validate that safety requirements are effectively preserved.

In the case of timing interference, it is particularly challenging to test key performance corners, such as interference caused by asynchronous activity (e.g., due to traffic arriving through an Ethernet port), especially if the type of transactions that should be tested cannot be induced synchronously by the cores, hence precluding the use of software tests to validate those cases.

[0] This work has been partially supported by the Spanish Ministry of Science and Innovation under grant PID2019-107255GB-C21 funded by MCIN/AEI/10.13039/501100011033.

Recently, the SafeTI traffic injector [1, 2] has been proposed to tackle this gap. The SafeTI is capable of producing programmable traffic patterns, with high flexibility and controllability, hence allowing to test scenarios synchronously despite the fact they would only occur asynchronously in practice. Unfortunately, the current implementation of the SafeTI only works with AMBA Advanced High-performance Bus (AHB) interfaces, has been proven only in a 4-core homogeneous RISC-V multicore [3], and has some limitations to inject traffic at a sufficiently high frequency.

This paper introduces the enhancements performed in the SafeTI to remove most of its constraints, such as pipelining its architecture for higher injection frequency, porting it to the popular AMBA Advanced eXtensible Interface (AXI), and integrating it in multiple interfaces of a more powerful heterogeneous MPSoC based on Frontgrade Gaisler technology for the space domain [4]. We also provide some future prospects for this component.

SafeTI Enhancements

Overview. The SafeTI is a programmable traffic injector consisting of a traffic descriptor buffer (see Figure 1, top), a set of control registers, and the traffic injector itself, which consumes the descriptors from the buffer as dictated by the control registers that, for instance, indicate when the traffic injection should start. Traffic descriptors encode the traffic pattern to be generated, including a target address, whether the access is read or write, the amount of data to transfer, and repetitions. The original realization of the SafeTI has some limitations that we have removed. The remaining of this section describes the main ones.

Initialization of the descriptors. Originally, the SafeTI was connected to the AMBA AHB for

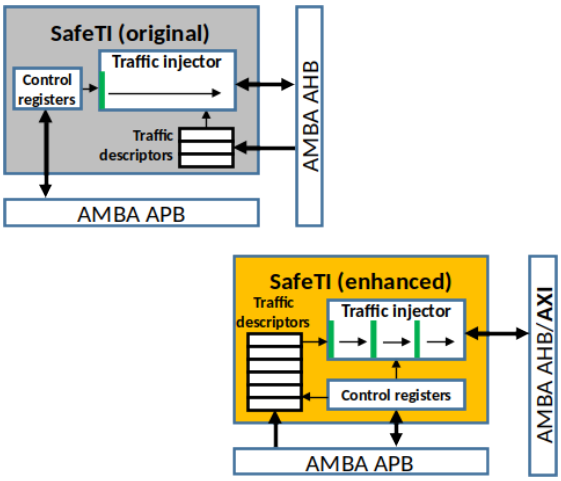

Figure 1: *Original SafeTI architecture (top), and enhanced one (bottom).*

both traffic injection and configuration (see Figure 1, top). However, programming descriptors in the buffer through the AMBA AHB data interface brought some concerns: (i) buffer updates could experience and cause timing interference, (ii) buffer updates could pollute cache memories, and (iii) buffer updates could become simply wrong.

Regarding timing interference, fetching descriptors through the same interface used for data transfers in the cores could create unforeseen interference in the cores. Regarding cache pollution, since descriptors were fetched through the regular data path, they could pollute cache memories evicting useful contents. Finally, if the SafeTI did not have access to coherent data from the cores, the descriptors would not be available for the traffic injector to load, retrieving obsolete contents from memory instead of the correct traffic pattern. To solve these limitations, we have enabled the use of the AMBA Advanced Peripheral Bus (APB) for configuration purposes. By being a separate interface, it neither interferes with data transfers, nor pollutes caches, nor experiences coherence problems since the APB interface writes coherent data to any SafeTI module. This is illustrated in Figure 1, bottom.

Injection rate. The original design of the SafeTI was not pipelined, which implied that some cycles were needed from the time a descriptor was fetched until the corresponding traffic was injected. This characteristic limited the injection rate since back-to-back descriptors needed to be fetched and decoded prior to execution. To solve this limitation, we have pipelined the design of the SafeTI, hence parallelizing descriptor fetch, decode, and execution and enabling sustained traffic limited by the interface only.

Target interfaces. Originally, the SafeTI was developed to support AMBA AHB only, since that was the protocol of the interface where it was first integrated [1, 3]. However, the architecture of the

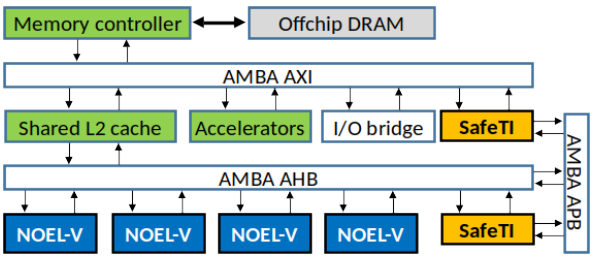

Figure 2: *SELENE SoC with 2 SafeTI modules.*

SafeTI is not restricted to any particular interface and, in principle, could be tailored to operate with virtually any interface. In our case, we have already extended it to work with AMBA AXI interfaces, such as those of the SELENE SoC [4], which is a RISC-V based MPSoC based on Gaisler's technology and intended for domains such as space, railway, and automotive. This is illustrated in Figure 2, where we can see a schematic of the SELENE SoC with a SafeTI instance attached to the AHB interface and another to the AXI interface.

Future Plans

Our future prospects for the SafeTI include performing a more exhaustive verification and validation with the aim of easing its adoption, and devising appropriate descriptors for an exhaustive test campaign of the SELENE SoC as an illustrative example of the use of the SafeTI. In the mid-term, we also aim at tailoring the descriptors and, potentially, the SafeTI design itself to enable new applications in the area of security (e.g., to counteract side-channel attacks) and testing (e.g., test cache coherence protocols).

Summary

The SafeTI is a powerful tool to test performance corners with limited effort, but its original design had a number of limitations. This work shows that those limitations have been removed, and the current design is more efficient and portable.

References

[1] O. Sala et al. "SafeTI: a Hardware Traffic Injector for MPSoC Functional and Timing Validation". In: *IOLTS*. 2021. DOI: 10.1109/IOLTS52814.2021.9486689.

[2] BSC. *SafeTI Traffic Injector*. https://gitlab.bsc.es/caos_hw/hdl_ip/bsc_safeti/. 2021.

[3] N.J. Wessman et al. "De-RISC: the First RISC-V Space-Grade Platform for Safety-Critical Systems". In: *2021 IEEE Space Computing Conference (SCC)*. 2021, pp. 17–26. DOI: 10.1109/SCC49971.2021.00010.

[4] H2020 SELENE consortium. *SELENE RISC-V open source hardware platform*. https://gitlab.com/selene-riscv-platform. 2021.