HW-SW Interface for RAS in RISC-V Architectures

Daniele Rossi*, Nicasio Canino, Stefano Di Matteo and Sergio Saponara

Department of Information Engineering, University of Pisa, Italy

Abstract

System Reliability, Availability and Serviceability (RAS) are major properties in any High Performance Computing systems. An essential feature to improve RAS is represented by facilities to log hardware errors information and report it to system software. We present here the main characteristics of a HW-SW interface that we have developed to improve RAS of RISC-V architectures. We describe the main building blocks of the proposed peripheral that can be adopted in both 32- and 64-bit RISC-V architectures, their main characteristics, and the set of registers that are used to log and report main error information. Finally, an FPGA-based test platform developed to validate the proposed RAS peripheral is briefly discussed.

Introduction

The reliability, availability and serviceability (RAS) of a computing system can be increased with low impact on performance and cost through the addition of information and control path redundancy along with dedicated error-checking hardware [1]. If errors can be detected and corrected by error checking hardware, system operation can continue without any noticeable loss in performance [1].

Additionally, error containment limits the propagation of an erroneous data. This enhances system availability by limiting the effects of errors to a subset of software or hardware resources. System software may either correct the error and resume the interrupted program or terminate software processes that cannot continue due to the error. Error logging and reporting enhances RAS by providing information that is used by the software to identify erroneous blocks [2, 3, 4]. Consequently a hardware-software (HW-SW) interface for error logging and reporting is essential to improve RAS of a computing system. It defines the facilities by which hardware errors are logged and reported to system software using several banks of registers to record error information. This way system software can take actions to recover from and diagnose hardware errors. Therefore, the combination of hardware error detection and correction together with a HW-SW interface for error logging and reporting is mandatory to improve the RAS of a computing system.

So far, no HW-SW interface for RAS has been developed for RISC-V processors. RISC-V community has established RERI (RAS Error-record Register Interface) task group [5] that is in charge of identifying the main features to be implemented by such RAS interface. To fill this gap, within the European Processor Initiative (EPI SGA2), we have developed and implemented a HW-SW interface for error logging and reporting (*RAS Peripheral*) to be adopted in future HPC systems based on RISC-V processors. The RAS Peripheral has been developed by using System Verilog, synthesized with a 7 nm standard cell library for area occupation evaluation, and finally implemented in a FPGA-based test platform for verification and validation. All main features have been tested for and the provision of the expected services has been verified.

Developed RAS Peripheral

The developed RAS Peripheral has been designed according to the following error classification criteria.

- *Corrected Errors* (CEs), which includes errors that are detected and corrected by hardware.
- *Deferred Errors* (DEs), including detected uncorrected errors that have no immediate impact on the operation of the system. Operation can continue and dealing with the error is deferred to a later point in time if the corrupted data are consumed.
- *Urgent Errors* (UEs), consisting of detected errors that require immediate action from system software.

A high-level block description of the architecture of the developed RAS peripheral is depicted in Figure 1. It comprises the following main building blocks.

The **Error Mux** receives the error control signals generated by the error-checking circuitry of the monitored blocks and generates an error message comprising a Source ID, which identifies the block generating the error, and Error Type (either CE, or UE, or DE).

The **FIFO Buffer**, a circular buffer including the Synchronization Buffers, which stores the error messages from the Error Mux and pair them with the address of the erroneous location coming from the Address Buffer.

The **Main Controller**, which selects the register bank where to locate a new error record. For efficiency purposes, it adopts a fully associative cache-like approach, selecting the first free bank that it finds. In case all banks are occupied by valid error records, it may overwrite one of them or discard the new error record and keep the older ones, according to the following hierarchy: UE > DE > CE.

* Corresponding author: daniele.rossi1@unipi.it. This work has been partially supported by EU HPC-EPI-SGA2 project and by the Italian Ministry of University and Research (MUR) in the framework of the FoReLab project (Departments of Excellence).

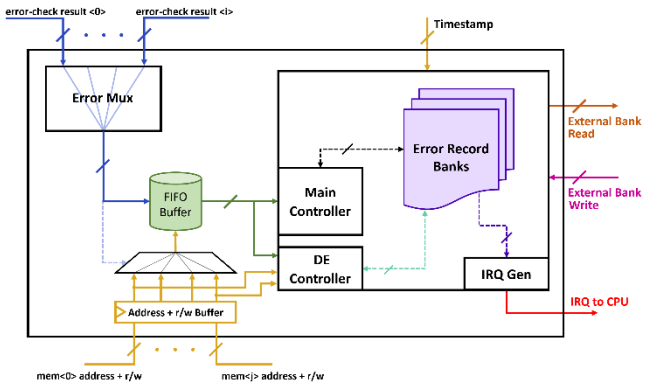

Figure 1. Architecture of the developed RAS peripheral.

The **DE Controller** handles the indication of deferred errors. To perform its task, the read/write signal is also necessary. Indeed, if a memory location storing data with a DE is accessed during a read operation, this poisonous data are going to be consumed. Therefore, the DE needs to be escalated to an urgent error UE. Instead, tracking a write operation to a location with a deferred error allows us to invalidate the corresponding error record, since in this case old data are going to be overwritten by new ones.

In the **Error Record Banks**, there are the following 64-bit registers to store all necessary information on errors.

Feature Register, which defines the main characteristics of the RAS interface and is configured at implementation stage. As an example, it specifies the type of errors that are logged (CE, DE, UE), whether the counter for corrected errors is implemented or not, if the timestamp is stored, kind of errors generating an interrupt, etc.

Control Register stores a set of enable signals for the different features (for instance, if the error reporting is enabled, for which cases the interrupt is enabled, etc). It can be modified at runtime.

Status Register stores information on the error, that is the error message, or error syndrome, and some valid bits related to the address and miscellaneous registers, if a corrected error counter overflow is implemented, or whether the error record has been overwritten.

Address Register stores the address of the erroneous data.

Two *Miscellaneous Registers*, which store additional information about the logged error. Its timestamp is also stored in this register.

Finally, the **Interrupt Generator** (IRQ Gen) is in charge of generating the interrupt signal in different scenarios, depending on the features of the RAS Peripheral chosen at implementation time (included in the *Feature Register*).

Table 1: Area occupation for 7nm standard cell.

Peripheral Blocks	Area (μm²)			Gate Equivalent		
	Comb	Seq	Tot	Comb	Seq	Tot
Err. Mux	3.52	4.84	8.36	45833	63021	108854
Synch. Buff.	0.17	27.42	27.59	2214	357031	359245
FIFO Buff.	73.78	79.49	153.27	960677	1035026	1995703
Err. Rec. Banks	60.85	87.97	148.82	792318	1145442	1937760
Controllers	86.48	3.88	90.36	1126042	50520	1176562
IRQ Gen.	1.94	0.40	2.34	25260	5209	30469

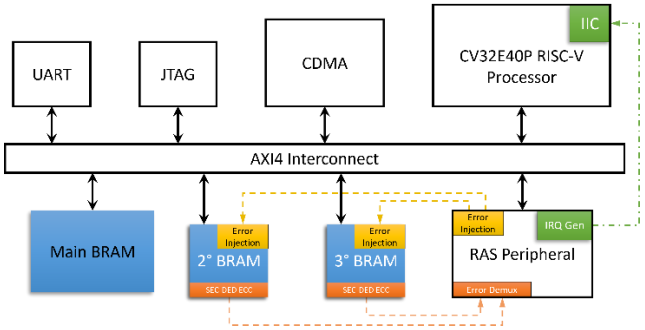

Figure 2. Test prototype for RAS Peripheral validation.

In order to evaluate area overhead of the developed RAS peripheral, we have synthesized it with a 7 nm standard cell technology. As an example, we have considered two register banks. Table 1 reports the area occupation figures.

FPGA-based Test Platform Implementation

The RAS Peripheral developed has been implemented and prototyped to verify and validate its behavior using an FPGA board Xilinx Zynq ZCU104. Figure 2 shows the structure of the prototype developed for testing purposes. All blocks are accessed via memory mapping, and AXI4 communication protocol is implemented. The RISC-V processor is a CV32E40P core, a 4-stage in-order 32-bit RISC-V processor that implements RV32IMFC ISA [6]. It is worth noting that the developed RAS Peripheral can be also adopted with 64-bit RISC-V processors. The implemented test platform includes a UART interface that is used to print on terminal, and a JTAG for code loading on Main Block RAM (BRAM), which is also used for data. Central Direct Memory Access (CDMA) soft Xilinx IP core is used for memory-to-memory transfers between second and third Block RAMs. These two memory blocks implement a SEC-DED ECC for single error correction and double error detection, allowing us to identify whether the block transferred includes any memory location with erroneous data. It is worth noting that ECC circuitry and fault-injection ability for Block RAMs is provided by Vivado. All performed tests have confirmed the correct behavior of the developed RAS Peripheral.

References

[1] R. Canal et al., "Predictive Reliability and Fault Management in Exascale Systems: State of the Art and Perspectives". *ACM Computing Surveys* 53(5), Jan. 2020.

[2] *AMD64 Architecture Programmer's Manual, Volume 2: System Programming*, Oct.. 2022.

[3] *Intel Xeon Processor E7 Family: Reliability, Availability, and Serviceability* - White paper.

[4] *Arm® Reliability, Availability, and Serviceability (RAS) Specification - Armv8 Architecture Profile*, July 2019.

[5] RERI (RAS Error-record Register Interface) task group. URL: https://lists.riscv.org/g/tech-ras-eri.

[6] RISC-V International. URL: https://riscv.org/.