COBHAM

Introduction of fault-tolerant concepts for RISC-V in space applications (RV4S)

Final Report

Report

2019-12-10

Doc. No RV4S-FR-0013

Issue 1.0

Contract 4000123876/18/NL/CRS

Deliverable FT

CHANGE RECORD

Issue	Date	Section / Page	Description
0.1	2019-12-09	All	First draft for project internal review.
1.0	2019-12-10		Updates after project internal comments

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope of the Document

This document establishes the final report of the "Introduction of fault-tolerant concepts for RISC-V in space applications (RV4S)" activity initiated by the European Space Agency under ESTEC contract 4000123876/18/NL/CRS.

Cobham Gaisler AB (SE) has together with Universitas Nebrissensis S.A. – ARIES Research Center (ES), and QinetiQ Space NV (BE) studied the open instruction set architecture RISC-V and how this open standard, and a specific implementation of the standard, can be applied in European space systems.

1.2 References

The following documents are referred as they contain relevant information:

[RD1] D1: RV4S-TN-00003-QS – RV4S User experience of integration of space processors in space equipment, Issue E, Revision 0, 2019-10-11

[RD2] D2: RV4S-PRVI-0002 – Technical Report: Proposed RISC-V Integration, Issue 1, Revision 2, 2018-11-06

[RD3] D3: RV4S-RHRV-0003 – Analysis of the hardening possibilities on RISC-V, Issue 1.2, 2018-12-04

[RD4] D4: RV4S-RP-00006-QS – RISCV End User Validation, End user validation report, Issue C, Revision 0, 2019-10-11

[RD5] D5: RV4S-DD-0005 – Design Description: Test Setup including VHDL and Board, Issue 2.0, 2019-12-06

[RD6] D6: RV4S-TPI-0006 – Test Plan: Integration Testing, Issue 1.0, 2019-06-11

[RD7] D7-9: RV4S-RVFI-0007 – RISC-V fault injection report, WP425, Issue 0.4, 2019-10-16

[RD8] D8: RV4S-TRI-0008 – Test Report: Integration Testing, Issue 2.0, 2019-12-09

2 ABBREVIATIONS

ESA	European Space Agency
ESTEC	European Space Research and Technology Center
RV4S	Introduction of fault-tolerant concepts for RISC-V in space applications
TBD	To Be Defined
GRLIB	Gaisler Research Library
BAR	Berkeley Architecture Research
RTL	Register Transfer Level
SOC	System-on-Chip
HDL	Hardware Description Language
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
TLB	Translation Lookaside Buffer
MMU	Memory Management Unit
OS	Operating System
DDR	Double Data Rate
DMI	Debug Module Interface
BEU	Bus Error Unit
SECDED	Single Error Correction Double Error Detection
PLIC	Platform Local Interrupt Controller
SEM	Soft Error Mitigation
FMC	FPGA Mezzanine Card
HPC	High Pin Count

3 BACKGROUND

Most European space-grade microprocessors today implement the SPARC instruction set architecture. An instruction set architecture defines the interface between software and hardware. Adhering to one architecture facilitates software reuse and reuse of design and engineering knowledge. SPARC was selected as a preferred architecture by the European Space Agency in the 1990's and today several implementations of SPARC microprocessors exist in devices such as SCOC3 (Airbus Defense and Space), EPICA-NEXT (Thales Alenia Space), AT697 (Atmel), UT699/UT799 (Cobham/Aeroflex), GR712RC and GR740 (Cobham Gaisler). The reasons behind ESA selecting the SPARC architecture included excellent software support (at the time, SUN Microsystems produced workstations based on SPARC), openness and that the architecture was free of limiting patents.

Over time, the European space-grade microprocessors have evolved into system-on-chip devices that include a processor core implementing the SPARC ISA (Instruction Set Architecture) and a large set of peripheral units in the same device. On the commercial side, the SPARC architecture now has fewer users with most development targeting proprietary architectures from Intel and ARM.

The dominance of in particular ARM in the commercial sector has fuelled the need for a new modern, open, and unrestricted instruction set architecture. In recent years this has surfaced as part of the RISC-V architecture that now shows a major momentum in adoption by companies such as Google, AMD, NVIDIA and HPE (Hewlett Packard Enterprise).

The RISC-V architecture offers several potential benefits for end users:

- Attractive license model (BSD open source license), enabling students, end users and auditors to become familiar with the details of the chip design. Eventually this will lead to enhancements and better products.
- Strong academic and industrial support which confirms that the RISC-V architecture is a good potential candidate for future processing alternatives.
- Low silicon floor space requirements resulting in more logic resources being available to peripheral functions and interfaces. This leads to higher integration density which is a significant advantage in spacecraft applications. In the end an improved system will be obtained, exhibiting higher reliability at a lower cost.
- The slightly larger memory size requirement typically imposed by RISC architectures can be easily compensated by larger memory capacities becoming nowadays available for use in spacecraft avionics equipment. In addition the expected gain on Silicon floor space (as stated in the previous bullet) will also compensate for this minor loss.

4 OVERVIEW OF THE ACTIVITY

The technical objectives of this work were:

- **O1**: Evaluate the current state of the RISC-V developments and how they can be applied to use in European space
- **O2**: Select one existing processor implementation of RISC-V and integrate it to a contemporary European space-grade system-on-chip
- **O3**: Evaluate radiation hardening techniques that should be applied to the selected microprocessor IP to make it suitable for use in the harsh space environment
- **O4**: Create a demonstrator design implemented on field-programmable gate array (FPGA) technology to allow software evaluation of the architecture

The programmatic objective of the work is to lower the threshold for adoption of bleeding edge commercial technology in the space sector and to provide a modern microprocessor alternative that comes with the same level of openness as SPARC. This in turn will make software advancements on the commercial side available to the space industry. The tangible end goal, the FPGA implementation, will demonstrate the feasibility of merging existing space industry system-on-chip architectures with an implementation of a modern instruction set architecture. The innovation and novelty of the proposal lies on the technical side. The concept of taking a successful open ISA from the commercial market and introducing it for space has already been done by industry and the agency with the SPARC architecture and we aim to start a repeat of this success story. The concept of taking a commercial implementation and hardening it has also been done before.

5 OVERVIEW OF WORK PERFORMED

The following major technical steps are were performed (with related objectives id referred in parenthesis):

- Evaluation - general (**O1**) and specific (**O2**)
- Evaluation of hardening techniques (**O3**)
- Integration of RISC-V core in Cobham Gaisler's GRLIB IP library (**O2**)
- Implementation of hardening techniques (**O3**)
- FPGA synthesis (**O4**)
- Evaluation on board (**O4**)

The work started with the evaluation work where an analysis of the RISC-V ecosystem was performed. Analysis results were described in *D1 Analysis of RISC-V: state of the art and current implementations [RD1]*. The project partners performed a survey of the ecosystem and then a comparison of the RISC-V implementations:

- Rocket,
- lowRISC
- SHAKTI
- cores from the Pulp project
- BOOM

It can be noted that the development of RISC-V cores moves very quickly and several new releases were announced only in the time between PDR delivery and PDR.

The goal of the integration work was to select a RISC-V implementation and adapt it for implementation in representative SoC on FPGA board. The implementation that was selected was the Rocket chip from UC Berkley, which can be considered a flagship project within the RISC-V community. The integration work was described in deliverable *D2 Technical Report: Proposed RISC-V Integration [RD2]*.

Figure 1 below depicts the SoC system implemented as part of the work.

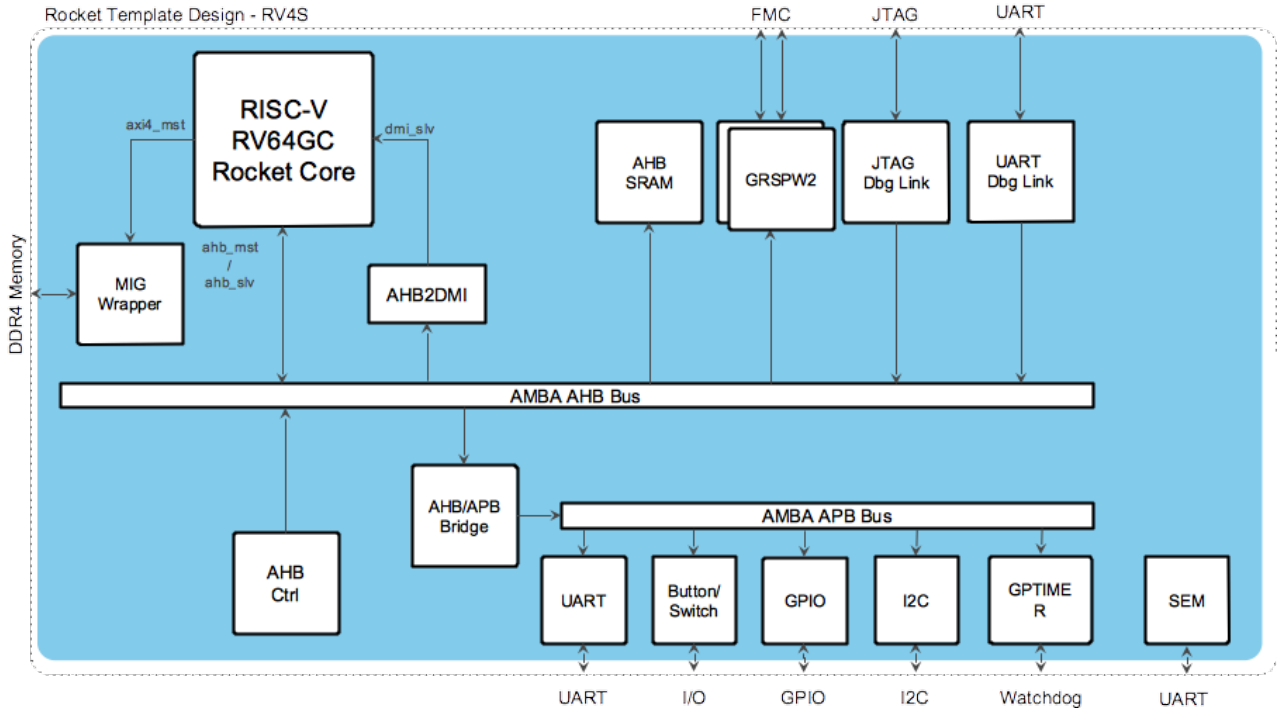

Figure 1: RV4S SoC Block Diagram

The RISC-V implementation, Rocket Core, was included in a system considered representative for contemporary European space-grade microprocessors (such as the Cobham GR712RC). The inclusion was made by only changing configuration options of the Rocket core, thereby enabling easy upgrades of the system to include new versions of the Rocket core. No changes to the Rocket source code were made. The included RISC-V processor was implemented with the following characteristics:

- A single RV64GC (short for RV64IMAFDC) RISC-V Rocket core
- 16 KiB of L1 4-ways Instruction and Data Cache with SECDED
- Multiplication and Division Hardware Unit with Early Out
- SV39 compatible MMU
- Double-precision Floating Point Unit
- Up to 31 External Interrupt Lines
- Bus Error Unit (BEU) for Cache and Bus Errors
- AXI4 Master Interface for main memory
- AHB Master and Slave Interfaces for the AMBA 2.0 bus
- A RISC-V Debug Module with a DMI Interface

The developed SoC system was subsequently implemented on a commercial FPGA development board, the Xilinx KCU105 FPGA development kit. Integration tests were performed and computational performance was evaluated using benchmark applications, please see section 6.

An integral part of the performed work was to introduce and evaluate hardening techniques for the RISC-V implementation. The work within this research activity was limited to the processor subsystem, which means that only the Rocket core part of the SoC was subjected to hardening measures.

At a first stage, the design was hardened by manually applying block-TMR (modification of the generated Verilog “netlist”). Variants were created using manual, or ad-hoc, methods:

- Basic data path protection with TMR, where TMR was applied to subcomponents of the Rocket core
- Ad-hoc technique, where area-efficient protection of the register file was obtained by using parity and sparing.

After technical discussions during review meetings, the work within the activity was then also expanded to explore usage of distributed TMR (DTMR) applied by synthesis tools, and this also became the mitigation measure of focus during subsequent error injection validation. The fault injection results are reported in [RD7].

In parallel with the work described above, Qinetiq space filled the role as a consumer and performed an end user evaluation that was ongoing throughout the activity. The end user evaluation work consisted of:

- Contribute with input based experience of integration of space processors in space equipment and review and give feedback to analysis and trade-off work.
- Perform a project internal evaluation of the proposed demonstrator system in terms of bus topology and selected peripherals.
- Perform an end user evaluation of the demonstrator system. The focus of the end user evaluation will be on the RISC-V processor implementation (not on the selected peripherals).

The end user evaluation results are provided in section 7.

The itemized list below summarises the work performed in relation to the technical objectives of the activity:

- O1: Evaluate the current state of the RISC-V developments and how they can be applied to use in European space
  - Survey of RISC-V IPs performed, submitted as analysis report
  - End user evaluation performed on reports and demonstrator developed within activity
- O2: Select one existing processor implementation of RISC-V and integrate it to a contemporary European space-grade system-on-chip

- Rocket-chip selected and integrated in GRLIB SoC design

- O3: Evaluate radiation hardening techniques that should be applied to the selected microprocessor IP to make it suitable for use in the harsh space environment
  - Manual TMR and Ad-hoc techniques implemented through modifications of the generated Verilog code
  - DTMR applied using Mentor Precision Hi-Rel on generated Verilog code
  - Evaluated using error injection in FPGA configuration memory
- O4: Create a demonstrator design implemented on field-programmable gate array (FPGA) technology to allow software evaluation of the architecture
  - Design implemented on Xilinx KCU105 development board with XCKU040 FPGA
  - Design validated and benchmarked.

6 BENCHMARK RESULTS

6.1 Performance

Performance measurements were performed as part of the integration tests and reported in [RD8]. Performance comparisons were also performed with LEON3FT SPARC systems and the results can be found in [RD8]. The performance comparisons show that the Rocket implementation outperforms LEON3FT at the expense of consuming a larger amount of logic resources (additional area required for ASIC implementations and a greater percentage of programmable logic device resource consumed). The work performed in the project did not provide an apples to apples comparison of the two processor models where both processor cores were instantiated with suitable memory subsystems and the same toolchain version were used to generate the benchmarks. The primary conclusion is that there is no reason to suspect that RISC-V implementations with current toolchains would have a problem delivering the same performance as contemporary space-grade processors.

The table below shows an excerpt of the benchmark results of the FPGA system running with a 100 MHz system clock.

Benchmark	Toolchain	Result	Comment
Coremark	Bare-C	232 CoreMarks	2.32 CoreMark/MHz
Dhrystone	RTEMS rvfs64imac	197.77 DMIPS	1.98 DMIPS/MHz
Whetstone	RTEMS rvfs64imac	24.4 MIPS	0.24 MWIPS/MHz
Linpack	RTEMS rvfs64imac	0.83 Mflops	0.83 Mflops
Dhrystone	RTEMS rvfs64imafd	204.66 DMIPS	2.05 DMIPS/MHz
Whetstone	RTEMS rvfs64imafd	209.5 MIPS	2 MWIPS/MHz
Linpack	RTEMS rvfs64imafd	8.12 Mflops	8.12 Mflops
Dhrystone	RTEMS rvfs64imafdc	198.64 DMIPS	1.99 DMIPS/MHz
Whetstone	RTEMS rvfs64imafdc	203.8 MIPS	2 MWIPS/MHz
Linpack	RTEMS rvfs64imafdc	8.74 Mflops	8.74 Mflops

From the results above we can see that the system achieves:

- Dhrystone: A Bare-C Dhrystone figure of 1.24 DMIPS/MHz and around 2 DMIPS/MHz for the RTEMS builds. Other sources report 1.72 DMIPS/MHz for Rocket [RD8].
- CoreMark: 2.32 CoreMark/MHz. The CoreMark figure is in line with other results reported for Rocket [RD8].
- Whetstone: 2 MWIPS/MHz for RTEMS benchmarks that makes use of the available hardware support.
- Linpack: Around 9 Mflops with hardware floating-point support and one tenth of the performance with soft-float. Reliable results for other Rocket implementations have not been found.

6.2 Code size

A comparison of code size for benchmark binaries was performed and reported in [RD8]. The table below lists the text and data segment for the main object files from Whetstone, Dhrystone and CoreMark (no support files for functions like UART). Comparing the full linked binaries for the benchmarks was shown to be misleading since the support files are varying in size for the two architectures.

Filename	RISC-V64G			RISC-V64GC			SPARC V8		
	text	data	bss	text	data	bss	text	data	bss
whetstone.o	1600	0	80	1346	0	80	1656	0	80
dhrystone_main.o	4211	0	10280	3879	0	10280	3781	0	10264
dhrystone.o	508	0	0	326	0	0	396	0	16
core_list_join.o	2160	0	0	1510	0	0	2116	0	0
coremark_main.o	3556	24	0	3028	24	0	2994	12	0
core_matrix.o	2348	0	0	1650	0	0	2060	0	0
core_portme.o	96	8	28	80	8	28	1921	0	0
core_state.o	2053	0	0	1555	0	0	1921	0	0
core_util.o	420	0	0	288	0	0	432	0	0
Linpack	5516	0	40	4356	0	40	5692	0	40

The difference between the two RISC-V variants is that RISC-V64GC makes use of compressed instructions. Options (relevant for code generation) used:

RV: -std=gnu99 -O2 -ffast-math -fno-common -fno-builtin-printf -mabi=lp64 -march=rv64g / rv64gc

SPARC: -std=gnu99 -O2 -ffast-math -fno-common -fno-builtin-printf -mcpu=leon3

7 END USER EVALUATION RESULTS

Conclusions for FPGA demonstrator developed within this ITI activity is that the supporting software tool chain and debugging capabilities require significant improvements to be production ready. In summary the following features need to be investigated:

- Missing processor reset signal from debugger
- Unexpected behaviour of some debugger commands
- Lack of instruction/AMBA bus tracing support
- Lack of full CSR support and missing documentation
- Missing exception handling/reporting in debugger
- Mismatch between hardware implementation pane and compiler for ulong/long word sizes

The overall conclusions for adoption of the RISC-V ISA were:

- The following RISC-V ISA extensions are considered necessary to support space software application development: (All extension met by Rocket implementation, the general purpose ‘G’ configuration of RISC-V)
  - M-extension: integer numbers multiply and divide
  - A-extension: read/write atomic operations for multi-processor systems
  - F-extension: single-precision floating-point operations
  - D-extension: double precision floating-point operations
- Current and near future ESA projects recommend system integrators to use the RTEMS operating system. A pre-qualification test suite is currently available to verify the compatibility with OBC hardware in which the GR712RC processor is used. New processing solutions will require to extend RTEMS with SMP capability. In case RTEMS remains to be used for space application software, an update of the pre-qualification test suite will be also needed.

The detailed technical conclusions are summarized with:

1. No major functional issues were identified during the end user evaluation activities.
2. RISC-V has a modern and extensible architecture, designed from the ground-up and according nowadays principles: multi-core, hypervisor, MMU.
3. No more debug trouble of using IU register windows (SPARC architecture).
4. A reliable standardized debug interface (with GDB) is considered essential. The capability to perform non-intrusive instruction tracing and local bus monitoring is considered an advantage and will be helpful to troubleshoot multi-core device configurations.
5. Potential to apply application level space partitioning (hypervisor) and process/task level

space partitioning (MMU) capability.

6. Processing performance compared to existing space qualified processing solutions is satisfactory for Integer Unit instructions and significantly improved for Floating Point instructions.
7. Bit manipulations are costly CPU cycles. Some frequently used function examples are CRC computation, endianness bit swap, extract/expand, leading/trailing zero count, minimum/maximum. These relatively simple operations are frequently used in low level software (device drivers) to control I/O operations. Combine / expand are also beneficial for memory bandwidth driven data processing applications (compress several sub-word bit-strings into one memory word). Typical for image pre-processing. The B-extension (bit manipulation instructions) would significantly enhance the low level software processing capability in future generation space processors.
8. Vector or SIMD operations (mostly floating point) are considered beneficial to support GNCC and other kinds of control loop applications.
9. The availability of a hypervisor (H-extension) is considered an important asset to enable deployment of application software developed by different organizations or developed using different criticality/quality level.
10. The G-Extensions (I + M + A + F +D) are considered essential.

8 DELIVERABLES

The deliverables in the table below have been produced within this contract. Please see section 1.2 for document references.

Ref	Title/Description
D1	Technical Report: Current status of RISC-V
D2	Technical Report: Proposed RISC-V Integration
D3	Technical Report: Hardening possibilities for RISC-V
D4	End User Validation Plan and Report
D5	Design Description: Test setup including VHDL and board
D6	Test Plan: Integration testing
D7/D9	Test Plan and Report: Validation testing
D8	Test Report: Integration testing
TDP	Technical Data Package
ESR	Executive Summary Report
AB	Abstract
FT	Final Report
CCS	Contract Closure Summary
WAT	Website Achievement Template
TAT	Activity Achievement Template
FP	Final Presentation
SW1	Regular RISC-V IP core
SW2	Protected RISC-V IP core, final
SW3	Test software and log data from all testing and validation
HW1	FPGA test platform

9 APPLICATION OF TECHNOLOGY DEVELOPMENTS RESULTING FROM THE ACTIVITY

The work performed in the activity is relevant in the long term for all space systems that utilize microprocessors. From a historic perspective, new technology as the one proposed here would first be a candidate for inclusion in instruments (payload processing) as opposed to platform applications. With the introduction of nanosats the proposed architecture would also be a candidate for handling all spacecraft functions.

The market share within the space industry is notoriously difficult to assess since the market is traditionally to a large extent created by international and national space agencies.

Several trends that can be identified are:

- There is traditionally a resistance to adoption of new technology for space, since inclusion of the new technology adds risk. The advent of “New Space” companies that, making use of commercial technology, now compete with the established space companies has forced a shift. Platform lifetimes are now considered to be in terms of two or three years instead of ten or twenty years. The more frequent design changes in platforms opens a window for introduction of new technology at a faster pace.
- There is a continuous effort towards higher integration where more functionality is included in one device. This leads to savings in power, mass and area.
- The introduction of multicore devices now allow one processor device to handle both payload and platform functions.
- The existing software tools used within the space market are well known and engineers within the industry are comfortable with the tools available for existing SPARC and PowerPC platforms. The introduction of multicore devices requires more from development tools and the commercial sector, due to its size, has a bigger availability of tools for the platforms used in the commercial sector.

With the adoption of the RISC-V architecture, the space sector would get access to commercial development tools from a plethora of vendors. As RISC-V is expected to have continued success in the commercial sector, the space industry will also be positioned to benefit from availability of operating system developments from other markets. The same arguments hold for an introduction of ARM and, to a lesser extent, ARC and MIPS architectures. The unique position of RISC-V and the open implementations is that, as with SPARC, any entity can adopt or develop their own implementation. This prevents a single source for microprocessor products. Moreover, being an open architecture, there is access to research know-how from many universities.

A technology roadmap for RISC-V introduction in space system follows below:

1. Selection of RISC-V IP and hardening against radiation effects of IP
2. Integration of IP with peripherals that provide interfaces and functionality required for space

systems

3. Implementation of technology demonstrator
4. Establish software environment (toolchains, debuggers, operating system support) of production quality for demonstrator system
5. Refinement of demonstrator architecture based on user feedback
6. Implementation of refined system in space-grade technology (ASIC or space-grade FPGA)
7. Product launch of RISC-V based system

Items 1 – 3 above were covered by the work performed within the activity.

10 CONCLUSION

RISC-V general conclusions:

- Survey of current RISC-V ecosystem performed.
- Extensions necessary to support space software application development identified:
  - M-extension: integer numbers multiply and divide
  - A-extension: read/write atomic operations for multi-processor systems
  - F-extension: single-precision floating-point operations
  - D-extension: double precision floating-point operations
- A number of instruction extensions are a key asset of RISC-V
- No register windows (as opposed to SPARC) is a desirable architectural feature
- Many (open-source) IPs available, but their stability is yet questionable.

Demonstrator implementation performed in this project:

- Integration of RISC-V implementation into a typical space SoC architecture successful.
- FPGA demonstrator of specific RISC-V implementation was limited in functionality but still provided insights on use of RISC-V ISA and allowed comparisons with LEON implementations.

As part of use of the demonstrator, two major shortcomings were identified as compared to the traditional LEON based space-grade microprocessors:

- The Rocket implementation does not support instruction and bus trace buffers. These are considered important features to support on-target debugging
- The Rocket implementation does not provide any means to dynamically control the L1 cache, the cache is always enabled.

The radiation hardening with automatic "DTMR" methods on a commercial reprogrammable FPGA is considered successful. Single-event fault injection shows that the failure rate is reduced by about an order of magnitude. Moreover, DTMR causes huge overheads and is slowing down the performance. 'Smart' hardening requires modifications, considered difficult to do with the Berkeley Chisel implementation. Some insights gained on practicality of applying hardening measures on Xilinx Kintex Ultrascale.

A conclusion that has been found applicable both to RISC-V implementation selection and radiation in general mitigation: An IP core coded in native HDL is preferable.