Introduction of Fault-Tolerant Concepts for RISC-V in Space Applications (RV4S)

Final Presentation

ESA/ESTEC contract 4000123876/18/NL/CRS

Activity Overview

- A step towards evaluating the use of the RISC-V instruction set architecture (www.riscv.org) for European space applications.
- 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 projects
  - 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 devices with an implementation of a modern instruction set architecture.
- Budget: 175 kEUR
- Activity duration: 16 months
- TO: Roland Weigand
- Consortium: Cobham Gaisler (SE), Universitas Nebrissensis S.A. – ARIES Research Center (ES), QinetiQ Space (BE)

Activity Overview

- Final presentation in two parts:
- Part 1 focusing on:
  - 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
  - O4: Create a demonstrator design implemented on field-programmable gate array (FPGA) technology to allow software evaluation of the architecture
- Part 2 focusing on:
  - 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
- Followed by summary / conclusions

Work Breakdown Structure

AO/1-8876/17/NL/CRS
ESA Innovation Triangle Initiative (ITI)

WPs for Cobham Gaisler - Developer	WPs for ARIES - Developer	WPs for QinetiQ – Customer
WP110 - Support to analysis of RISC-V Status	WP120 - Analysis of RISC-V: state of the art and current implementations	WP430 – End User Validation
WP210 - RISC-V Processor Integration	WP220 - Support to the RISC-V Processor Integration	
WP310 - Support to analysis of hardening possibilities	WP320 - Analysis of the hardening possibilities on RISC-V	
WP410 - Design and implementation	WP420 - Design support	
WP415 - Integration testing		
	WP425 – Validation	
WP510 - Project management	WP520 - ARIES sub-contractor management and liaison with prime contractor	

Work Performed: Analysis
WP110/WP120/WP430

- Analysis results described in D1 *Analysis of RISC-V: state of the art and current implementations*
- WP120 performed a survey of the landscape and then performed a comparison of the RISC-V implementations:
    - Rocket,
    - lowRISC
    - SHAKTI
    - cores from the Pulp project
    - BOOM

Analysis of RISC-V: state of the art and current implementations
WP120
Technical report [illegible]
Document number: [illegible]
Issue [illegible]
ESA Contract [illegible]
Deliverable: [illegible]
UNIVERSIDAD NEBRIJA

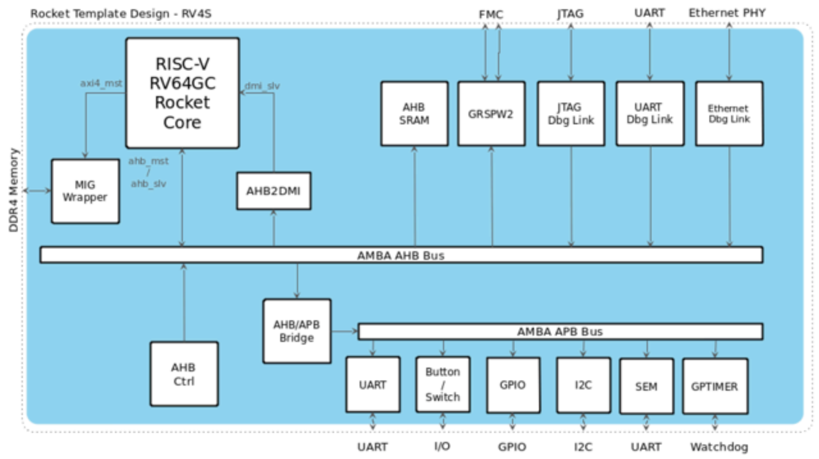
QINETIQ
RV4S
User experience of integration of space processors in space equipment

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

- Xilinx KC105 development board was selected and procured for the FPGA demonstrator platform.

Work Performed: Integration
WP210/WP220/WP430

- Goal of work was to select a RISC-V implementation and adapt it for implementation in representative SoC on FPGA board. Rocket chip implementation selected.
  - Rocket can be considered a flagship project. It is an SoC generator implemented in Chisel
  - A first implementation of the Rocket core in a GRLIB system had been delivered from CG to ARIES during the Analysis phase.
- Described in D2 *Technical Report: Proposed RISC-V Integration.*



COBHAM

Technical Report: Proposed RISC-V Integration

Technical Report 2018-11-06
Doc. No RV4S-PRVI-0002
Issue 1.2 Contract 4000123876/18/NL/CRS
Deliverable D02

QINETIQ

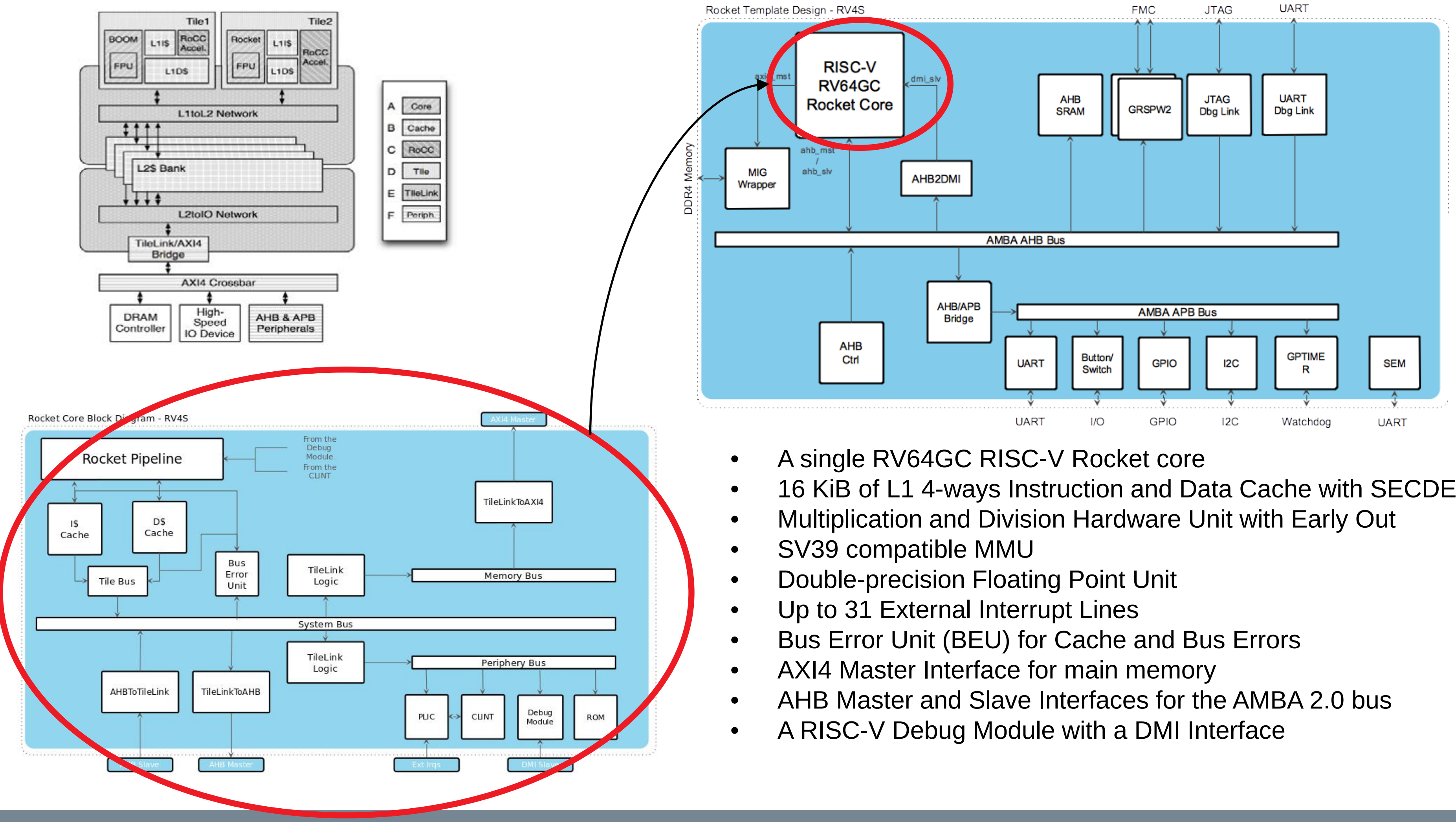
RV4S
Project internal evaluation of the proposed demonstrator system

Approved by	Date	Signature
Steven De Cuyper *Project manager*	13/11/2018	[illegible]
Piet Reuters *Product Assurance manager*	13/11/2018	[illegible]

Work Performed: Integration

WP210/WP220/WP430

- Rocket-chip integrated with GRLIB and implemented on Xilinx KCU105 board

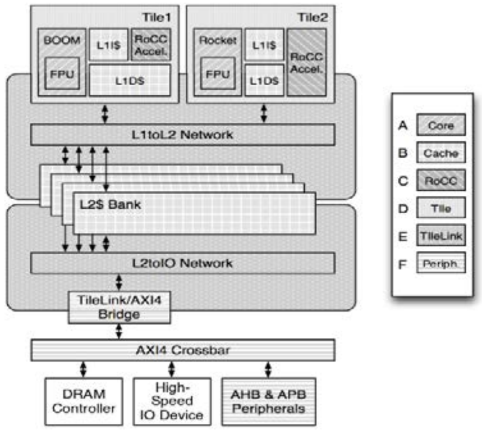


- A single RV64GC RISC-V Rocket core
- 16 KiB of L1 4-ways Instruction and Data Cache with SECDED
- Multiplication and Division Hardware Unit with Early Out
- SV39 compatible MMU
- Double-precision Floating Point Unit
- Up to 31 External Interrupt Lines
- Bus Error Unit (BEU) for Cache and Bus Errors
- 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

Work Performed: Hardening
WP310/WP320

- Effort to harden design described in *D3 Analysis of the hardening possibilities on RISC-V*.

- At a first stage, design was hardened by manually applying block-TMR (modification of generated Verilog “netlist”).
- Variants created using manual/ad-hoc methods:
  - Basic data path protection with TMR: TMR has been applied at Tile level
  - Ad-hoc technique: Efficient Protection of the Register File using parity and sparing



- Activity also explored usage of DTMR applied by synthesis tools – which was then the focus of the error injection campaign.
  - Today’s presentation will cover DTMR area results and error injection validation

Work performed: End User Evaluation

WP430

- End user evaluation consisted of:
  1. Contribute with input based experience of integration of space processors in space equipment and review and give feedback to analysis and trade-off work.
  2. Perform a project internal evaluation of the proposed demonstrator system in terms of bus topology and selected peripherals.
  3. 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).

End User Evaluation: Test cases

Test case ID	Characteristic	Goal	Note
1	Cross-compiler tool chain compatibility	Be able to compile and execute: i. Bare C Hello World application ii. RTEMS Hello World application	Bare-C and RTEMS toolchains functional. No GBD support. Bi-directional console communication partially supported.
2	Software upload and debugging capabilities	Be able to trace and step an application.	Could not verify back tracing, functionality not supported by demonstrator system. Demonstrator system provided limited debugging capabilities.
3	IU correlation against known reference	Be able to execute functionally correctly. Correlation will be verified against the SPARC V8 and MathWorks Simulink reference platforms.	ESA IBDM Control Software application was compiled and run successfully on the RISC-V processor. There are numerical differences between the outputs of RISC-V and LEON3 (on the order of 0.2% of the value).
4	FPU correlation against known reference	Be able to execute numerically correctly. Correlation will be verified against the ESA MLFS library.	ESA Mathematical Library for Flight Software (MLFS): compiles and works.
5	Performance benchmarks	Measure Dhrystone and Whetstone performance data.	Executed Dhrystone, Whetstone and CoreMark
6	Bit error injection/corrections	Be able to inject bit errors (SECDED) and correct/report in the caches and in the IU/FPU register file.	Not run. Interfaces not provided by Rocket. Error injection work in activity focused on configuration memory.

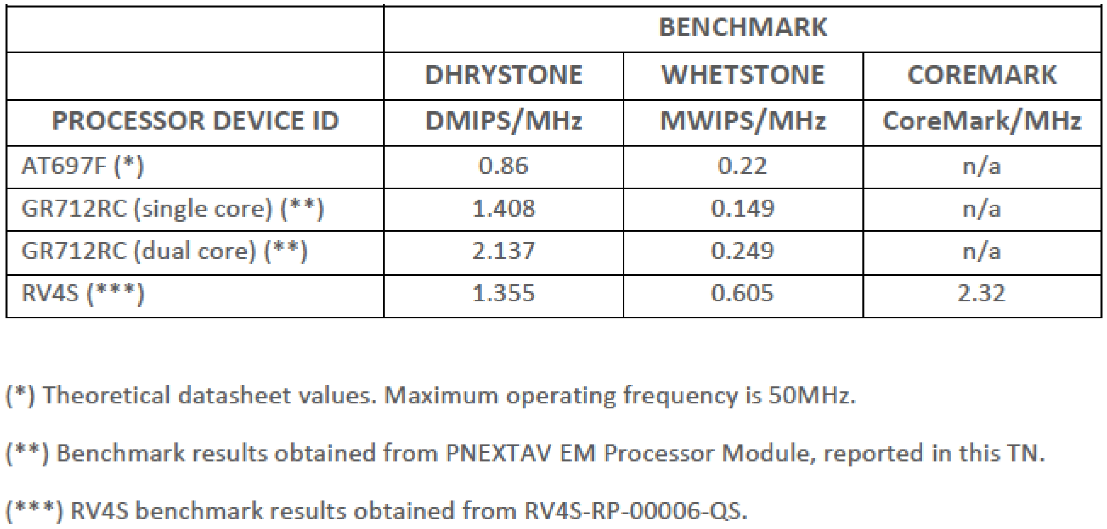
FPGA End User Evaluation: Conclusions

- Conclusions for FPGA demonstrator developed within this ITI activity:
  - 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
    - Differences observed in performance benchmarks ran by CG
- **Note:** Feedback above is for one specific implementation of Rocket, within a GRLIB system used via GRMON3 extended with various Tcl scripts.

Performance results

- **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.
  - Dhrystone reported for corresponding LEON3/4 systems: 1.4 – 1.9 DMIPS/MHz
- **CoreMark:** 2.32 CoreMark/MHz. The CoreMark figure is in line with other results reported
- for Rocket [RD4]. LEON4 achieves 2.05 CoreMark/MHz

Table 3 : Performance benchmark comparison (normalized per MHz)



	BENCHMARK		
	DHRYSTONE	WHETSTONE	COREMARK
PROCESSOR DEVICE ID	DMIPS/MHz	MWIPS/MHz	CoreMark/MHz
AT697F (*)	0.86	0.22	n/a
GR712RC (single core) (**)	1.408	0.149	n/a
GR712RC (dual core) (**)	2.137	0.249	n/a
RV4S (***)	1.355	0.605	2.32

(*) Theoretical datasheet values. Maximum operating frequency is 50MHz.

(**) Benchmark results obtained from PNEXTAV EM Processor Module, reported in this TN.

(***) RV4S benchmark results obtained from RV4S-RP-00006-QS.

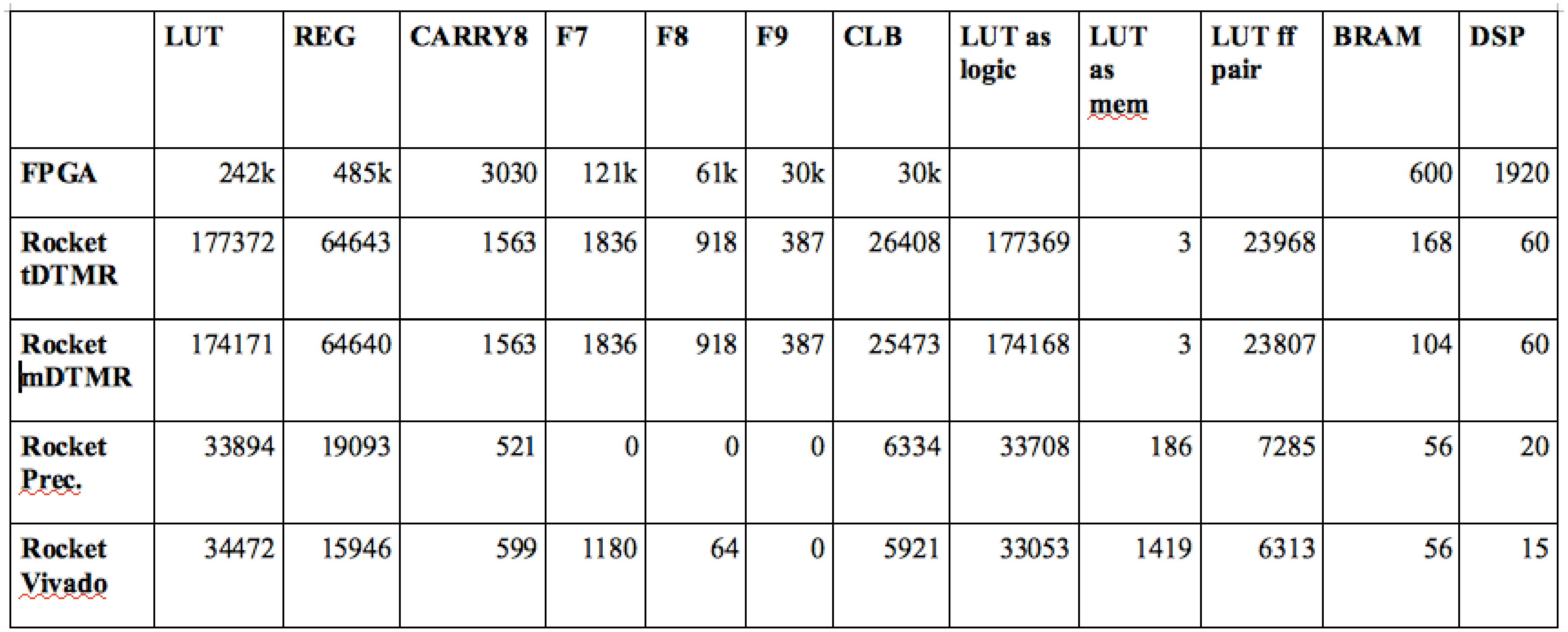
- Summary: Performance achieved between this Rocket implementation and general LEON3/4 is in line. Variation between cores is as big as variation between configurations.

Area comparison: LEON3FT vs Rocket

- Comparing the resource usage of the Rocket core with Leon3FT can be relevant if selecting between the two cores. From an implementation efficiency standpoint a comparison is of limited use since the CPU micro-architectures are very different.
- Besides obvious things like the Rocket being a 64 bit RISC-V and LEON3 a 32 bit SPARCv8, the Leon3FT only has static branch prediction.
- In our test builds, a Rocket without DTMR uses, compared to the Leon3FT, approximately
  - Half again as many LUTs
  - A third or so more REGs
  - Well over twice the CARRY8s
  - Nearly twice the BRAM
  - About the same number of DSPs
- This for a configuration that should be reasonably close regarding cache etc.
- A large difference in LUT, REG and CARRY8 use is expected since the integer cores are so very different. That the number of DSPs used is similar is unsurprising since that is mostly in the FPU, which is not affected by the integer core differences.
- For the same cache sizes, the LEON implementation makes more efficient use of bRAM (30 instances vs 56). bRAM usage for the Rocket has not been thoroughly investigated and it has been assumed that this is due to the mapping to bRAMs and not necessarily a requirement of the generic implementation.

Area comparison: Effects of DTMR

- In addition to the ad-hoc mitigation measures, DTMR was also evaluated. Examples of results:



	LUT	REG	CARRY8	F7	F8	F9	CLB	LUT as logic	LUT as mem	LUT ff pair	BRAM	DSP
FPGA	242k	485k	3030	121k	61k	30k	30k				600	1920
Rocket tDTMR	177372	64643	1563	1836	918	387	26408	177369	3	23968	168	60
Rocket mDTMR	174171	64640	1563	1836	918	387	25473	174168	3	23807	104	60
Rocket Prec.	33894	19093	521	0	0	0	6334	33708	186	7285	56	20
Rocket Vivado	34472	15946	599	1180	64	0	5921	33053	1419	6313	56	15

- "DTMR" means a build with DTMR added using Precision, but Vivado does the final steps. DON'T_TOUCH attributes are added. mDTMR - triplication of cache RAM disabled. tDTMR- using TCL DONT_TOUCH on whole triplication.
- "Prec" without TMR. "Vivado" means that Precision was not involved at all.

Code Size: RISC-V vs SPARC

Text and data segments for Whetstone, Dhrystone and CoreMark

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	140	8	28
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

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

Note: Benchmark performance differenced observed between G and GC but not further analysed.

End User Evaluation Conclusions (0/2)

Use of processors implementing the RISC-V ISA in space applications

- The following RISC-V ISA extensions are considered necessary to support space software application development: (All extension met by Rocket implementation)
  - 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.

End User Evaluation Conclusions (1/2)

Use of processors implementing the RISC-V ISA in space applications

COBHAM

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:
    1. Dhrystone benchmark: 1.36 DMIPS/MHz (QinetiQ Space Bare-C measurement)
    2. Whetstone benchmark: 0.61 MWIPS/MHz (QinetiQ Space Bare-C measurement using hard float)
    3. CoreMark benchmark: 2.32 CoreMark/MHz (Cobham-Gaisler Bare-C measurement)

End User Evaluation Conclusions (2/2)

Use of processors implementing the RISC-V ISA in space applications

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. The B-extension (bit manipulation instructions) would significantly enhance the low level software processing capability in future generation space processors. For more details see (https://raw.githubusercontent.com/riscv/riscv-bitmanip/master/bitmanip-0.90.pdf
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.

Work performed: Validation

- Following the analysis and integration work. ARIES, in parallel with the QinetiQ Space end user evaluation, performed an error injection campaign to evaluate the implemented hardening measures.
- Error injection campaign covered unprotected, manual triplication and DTMR. Results presented today cover unprotected design and DTMR.

Introduction of Fault-Tolerant Concepts for RISC-V in Space Applications (RV4S)

Fault Injection Validation

ESA/ESTEC contract 4000123876/18/NL/CRS

Luis Alberto Aranda
Juan Antonio Maestro

Agenda

1. Motivation
2. Experimental set-up
3. Fault injection results
4. Conclusions

Agenda

1. **Motivation**
2. Experimental set-up
3. Fault injection results
4. Conclusions

Motivation

RISC-V

open source initiative®

Motivation

- **FPGAs** error model is different from ASICs: configuration memory
- Alternatives to measure reliability of RISC-V:
  - **Radiate the device**: high costs
  - **Fault injection emulation**: high runtimes
- We have developed a tool (ACME - *http://www.nebrija.es/aries/acme.htm*) to pinpoint fault injection in specific areas of the RISC-V to reduce runtime

Agenda

Experimental set-up: Materials

KCU105 Evaluation board with
Kintex UltraScale FPGA

USB-to-UART PMOD

PC running MATLAB

Experimental set-up: Tools

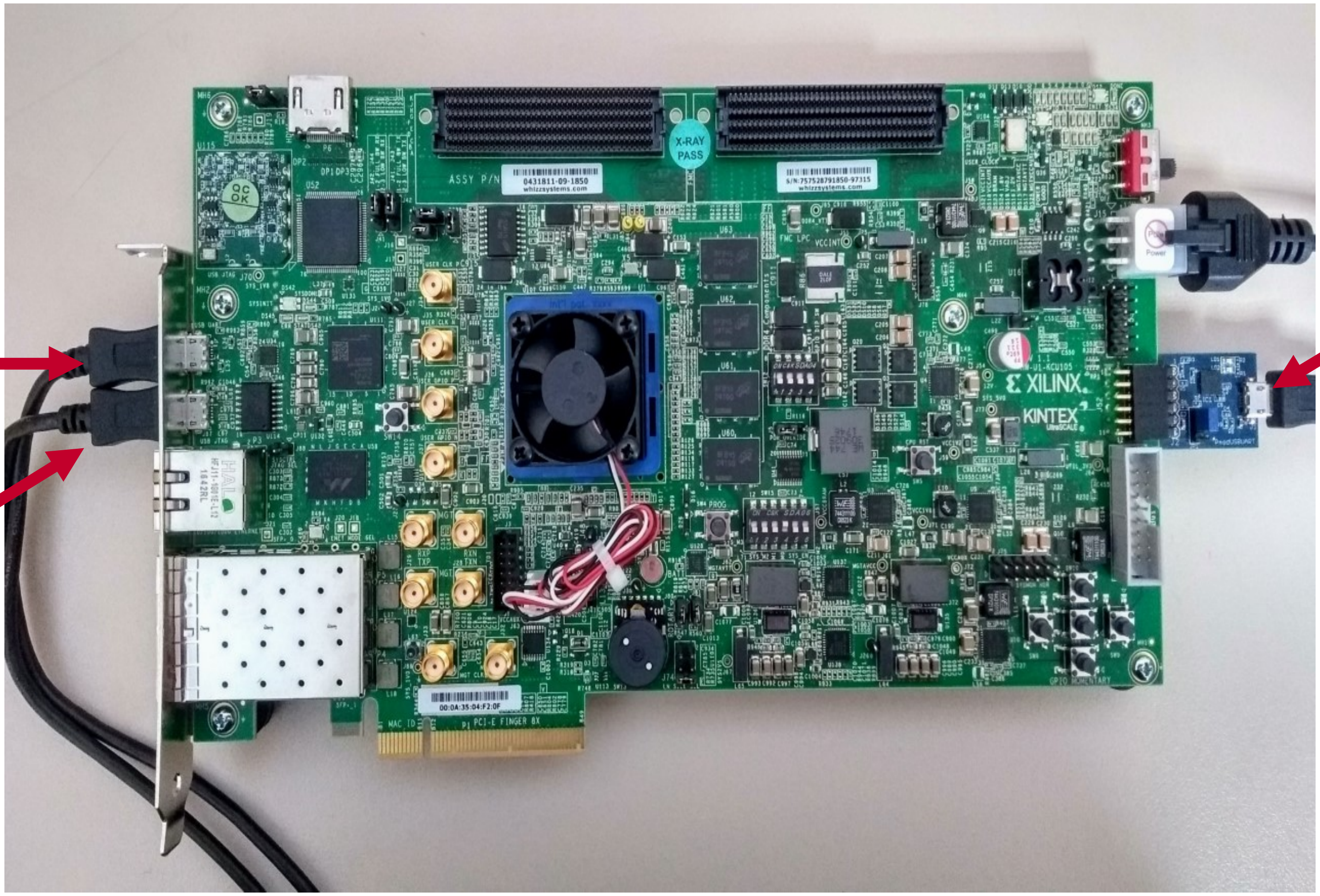
- **SEM IP**: Xilinx Soft Error Mitigation IP Controller
  - IP core developed by Xilinx
  - Enables error injection in the configuration memory of the FPGA
- **ACME**: Automatic Configuration Memory Error-Injection Tool
  - Configuration memory management tool developed by ARIES
  - Provides the injection addresses for the SEM IP
- **GRMON3**
  - Hardware monitor developed by Cobham Gaisler
  - Enables debugging and execution of benchmarks in the RISC-V processor

Experimental set-up: Benchmarks

Four synthetic benchmarks have been used:

- **Dhrystone**: integer benchmark to measure computing performance
- **Systest**: integer benchmark. Checks the status of the processor
- **Whetstone**: floating point counterpart of the Dhrystone benchmark
- **Linpack**: floating point benchmark. Performs several linear algebra operations

Experimental set-up: Overview



Benchmark output

GRMON control

SEM IP control

Experimental set-up: Fault injection procedure

The MATLAB script executes the following steps for each benchmark:

1. A bit-flip is injected at a random frame address with the SEM IP
2. The processor is initialized and the selected benchmark executed
3. The result of the benchmark is received and logged in a text file
4. The injected error is corrected with the SEM IP

Steps are repeated until the desired number of configuration bits is tested

Agenda

1. Motivation
2. Experimental set-up
3. **Fault injection results**
4. Conclusions

Fault injection results: Classification

- The benchmark is executed in the absence of errors to compare this **golden** result with the results obtained after the injection campaigns

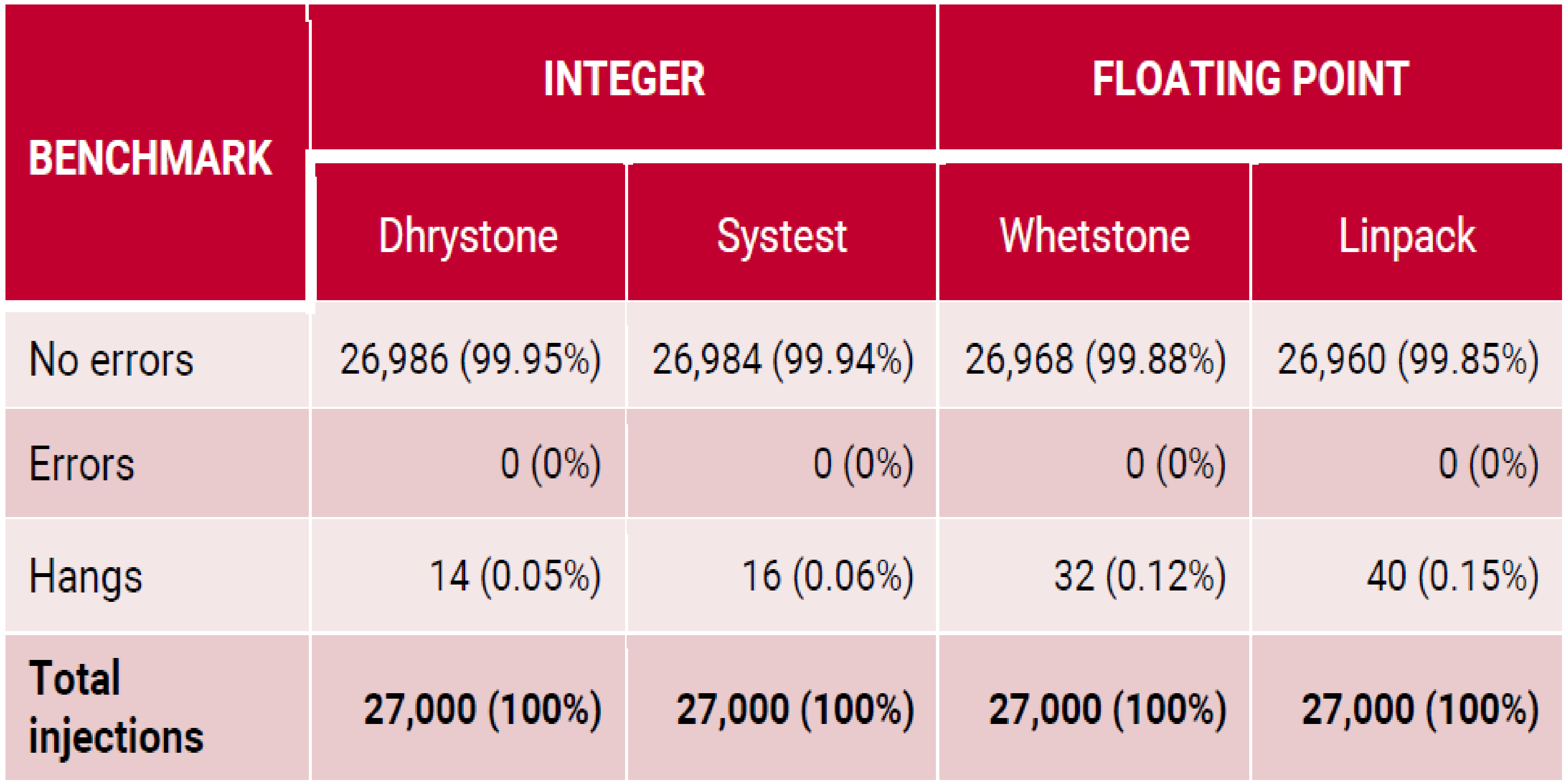
- The following classification has been done:

  - **No error**: the outcome of the benchmark matches the golden outcome
  - **Error**: the outcome of the benchmark does not match the golden outcome
  - **Hang**: there is no outcome or the execution of the benchmark never ends

Fault injection results: Unprotected RISC-V design

BENCHMARK	INTEGER		FLOATING POINT	
	Dhrystone	Systest	Whetstone	Linpack
No errors	26,162 (96.90%)	26,297 (97.40%)	26,025 (96.39%)	24,921 (92.30%)
Errors	329 (1.22%)	220 (0.81%)	494 (1.83%)	1,213 (4.49%)
Hangs	509 (1.88%)	483 (1.79%)	481 (1.78%)	866 (3.21%)
Total injections	**27,000 (100%)**	**27,000 (100%)**	**27,000 (100%)**	**27,000 (100%)**
Execution time	**7.5 seconds**	**7.2 seconds**	**12 seconds**	**15.4 seconds**

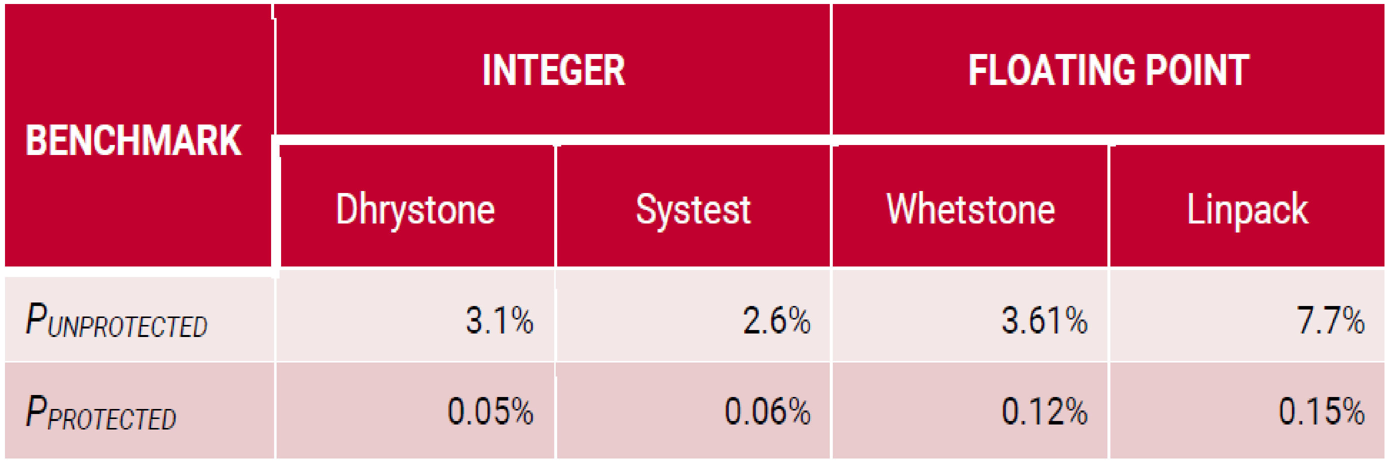
Fault injection results: DTMR-protected RISC-V design



BENCHMARK	INTEGER		FLOATING POINT	
	Dhrystone	Systest	Whetstone	Linpack
No errors	26,986 (99.95%)	26,984 (99.94%)	26,968 (99.88%)	26,960 (99.85%)
Errors	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Hangs	14 (0.05%)	16 (0.06%)	32 (0.12%)	40 (0.15%)
Total injections	**27,000 (100%)**	**27,000 (100%)**	**27,000 (100%)**	**27,000 (100%)**

Fault injection results: Fault tolerance comparison

- Probability of undetected errors in each design



BENCHMARK	INTEGER		FLOATING POINT	
	Dhrystone	Systest	Whetstone	Linpack
$P_{UNPROTECTED}$	3.1%	2.6%	3.61%	7.7%
$P_{PROTECTED}$	0.05%	0.06%	0.12%	0.15%

These values do not consider the total number of configuration bits

Fault injection results: Fault tolerance comparison

- Undetected errors per unit of time (#*U*):

$$\boxed{\#U = \phi \cdot \sigma \cdot N \cdot P}$$

- Where:

ϕ is the flux

σ is the cross section of the UltraScale FPGA

N is the number of configuration bits

P the probability of undetected errors

Fault injection results: Fault tolerance comparison

- For example:

$\phi = 1.00 \cdot 10^7 cm^{-2}/s^{-1}$ for the South Atlantic Anomaly

$\sigma = 1.87 \cdot 10^{-15} cm^2/bit$

$N_{UNP} = 6{,}665{,}452$ and $N_{DTMR} = 23{,}045{,}386$ bits

P obtained from the previous table

Fault injection results: Fault tolerance comparison

- Number of undetected errors per unit of time for unprotected and DTMR

BENCHMARK	INTEGER		FLOATING POINT	
	Dhrystone	Systest	Whetstone	Linpack
#U_{UNPROTECTED}	$3.86 \cdot 10^{-3}$ undetected errors/s	$3.24 \cdot 10^{-3}$ undetected errors/s	$4.5 \cdot 10^{-3}$ undetected errors/s	$9.6 \cdot 10^{-3}$ undetected errors/s
#U_{PROTECTED}	$2.15 \cdot 10^{-4}$ undetected errors/s	$2.56 \cdot 10^{-4}$ undetected errors/s	$5.17 \cdot 10^{-4}$ undetected errors/s	$6.46 \cdot 10^{-4}$ undetected errors/s

Agenda

1. Motivation
2. Experimental set-up
3. Fault injection results
4. **Conclusions**

Conclusions

- The unprotected design does not always fail:
  - There are errors that modify the outcome of the benchmark and errors that halt the processor
- The protected design does not always work:
  - There are errors that halt the processor, but this percentage is negligible
- The type of benchmark executed (integer vs. floating point) affects the percentage of errors obtained

Summary of work performed

- 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 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 benchmark comparison with existing space processors.

Problems Encountered

COBHAM

- Activity planning did not allow enough time for research on error mitigation measures and error injection tests
  - Development of FPGA prototype design was accelerated and delivered at PDR instead of CDR
- Problems getting the DDR4 SDRAM controller up and running in the KCU105 design
  - First versions of design used only on-chip memory while the DDR4 interface implementation was debugged
- Mitigation techniques were considered difficult to apply on Rocket Chisel description
  - Modifications were instead performed on generated Verilog code
- Power failure of FPGA development platform after lapse of warranty period.
  - Xilinx graciously provided additional board to support the ESA activity
- End user evaluation test case required GDB, which was lacking for RISC-V in GRMON
  - Introduce separate debug port and supply GBD connection through OpenOCD

Conclusions

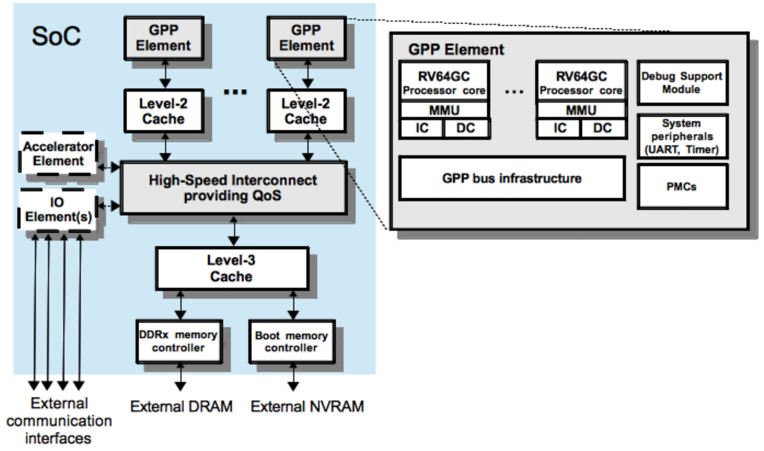
- RISC-V general conclusions:
  - Survey of current RISC-V ecosystem performed.
  - Extensions necessary to support space software application development identified.
  - A number of instruction extensions are a key asset of RISC-V
  - No register windows (as opposed to SPARC)
  - Many (open-source) IPs available, but their stability (maturity?) 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 RV implementation was limited in functionality but still provided insights on use of RISC-V ISA and allowed comparisons with LEON implementations.
  - Benchmark results comparable with existing LEON systems
- Radiation hardening with automatic "DTMR" methods considered successful, but huge overheads.
  - 'Smart' hardening requires modifications, considered difficult to do with the Berkeley Chisel implementation.
  - Some insights gained on practicality of applying hardening measures on XCKU
- Conclusion applicable both to RISC-V implementation selection and radiation in general mitigation: An IP core coded in native HDL is preferable.

Related Activies

H2020 De-RISC

Dependable Real-time Infrastructure for Safety-critical Computer

- European Commission Fast Track to Innovation Initiative
- De-RISC is a project that will productize a multi-core RISC-V system-on-chip from Cobham Gaisler AB and to port the XtratuM hypervisor from fentISS SL to that design, to create a complete computer platform consisting of hardware and software.
- Project partners:
  - Cobham Gaisler AB
  - FentISS SL
  - Barcelona Supercomputing Center
  - Thales Research and Technology
- More information: http://www.derisc-project.eu



De-RISC – Dependable Real-time Infrastructure for Safety-critical Computer

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement EIC-FTI 869945
www.derisc-project.eu