The Resilience Concept Validation by HiPeRCAR

Pasquale A. Marra [1] (pasquale.marra@thalesaleniaspace.com)
Daniel Akuatse [2] (daniel.akautse@syderal.ch)
Emily Crudo [3] (emily.crudo@galileoavionica.it)
Patrizia Bologna [3] (patrizia.bologna@galileoavionica.it)
Sergio Montenegro [4] (Sergio.Montenegro@dlr.de)
Raffaele Vitulli [5] (raffaele.vitulli@esa.int)

[1] *Thales Alenia Space Italia, SS Padana Superiore, 290 - 20090 Vimodrone (MI), Italy*
[2] *Syderal S.A., Neuenburgstrasse 7 - 3238-Gals (BE), Switzerland*
[3] SELEX - *Galileo Avionica SpA, Via Montefeltro, 8 - 20156 Milano, Italy*
[4] *FIRST, Kekuléstraße 7 - 12489 Berlin, Germany*
[5] *ESA/ESTEC, Keplerlaan 1, Postbus 299 - 2200 AG Noordwijk, The Netherlands*

ABSTRACT

The ESA-funded project 'HiPeRCAR' presents a possible implementation of resilient system, like the one designed at Thales Alenia Space Italia.
HiPeRCAR combines reliability and performance in a complex system able to react to failure events which can disturb a nominal work-plan.
The HiPeRCAR system was already presented at previous DASIA editions; the present paper completes those presentations with a critical analysis of the test results achieved with a real implementation running in a possible Space scenario.

1. INTRODUCTION

Present-day robotic controllers require high degree of processing power together with reliability and autonomy in order to cope with the needs of the new Space Missions.

HiPeRCAR interprets the ESA "*MOSREM*" concept [1], which places radiation-hardened nodes as front-end of a pool of high-performance shear nodes based on industrial processors.

So, a hybrid system has been designed where a reliable computer improves its processing capability by the help of high-performance computers: the front-end computer ensures the reliability, while the back-end computers provide the wanted processing power [2].
Reliability is ensured by the radiation-hardened technology, instead power-processing is provided by commercial computers (Fig 1).

The modular design eases the extension of the system: other nodes can be added for implementing redundant architectures or for controlling specific devices. The scalable and open architecture allows face the needs of different Missions and to fit with the financial resources of different Missions.

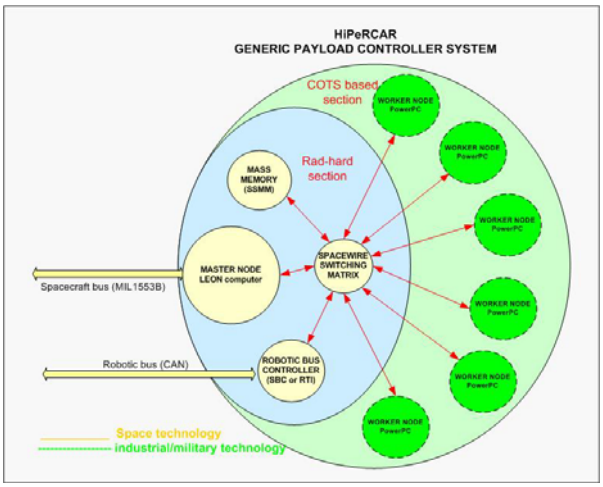

Fig. 1 – HiPeRCAR Conceptual System

By mixing Space technology and industrial technology, HiPeRCAR gives an answer to robotic or scientific applications requiring massive computations, or Space missions requiring Autonomy.

1.1 The HiPeRCAR Project

A consortium of European companies lead by Thales Alenia Space Italia designed such a system, trying to demonstrate that the resilience goal can be achieved.

The project requirements (ESA contract ITT/1-4607/04/NL/AG) impose also some challenging constraints as:
- use of European technology
- use of COTS (as alternative to developed) products
- use of open-source Software
- design at minimum cost.

A first design phase provided the major features of a possible system for Space; its natural role is as payload controller. The architecture of HiPeRCAR system was presented at DASIA 2006 [3].

Then, the project continued realising a target model that emulates a Space system. Also a SW Simulator of this target has been realised in order to assess the Software architecture of this resilient system. The Simulator allowed to design the robotic tasks and to confirm the architectural design [DASIA 2007 [5].

Now, the present paper deals with the third phase of the project that is it shows the results measured on the target. So, it is possible to do a critical analysis of the design and to confirm the resilient architecture.

2. HiPeRCAR TARGET DEMONSTRATOR

The hardware design is kept intentionally at low expenses considering as requirements only the major functionalities of a Space system.

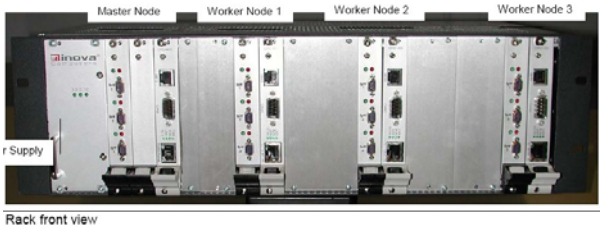

Fig. 2-1 – HiPeRCAR TARGET

The elements of the HiPeRCAR system are:

- ***Master node***

The rad-hard computer is made of a LEON processor. Here, it is substituted by a cheaper PowerPC processor (the 'PPC405' commercial board by ESD Electronic has been adopted for the full system). The board reliability could be improved at Military qualification degree.
The Command & Control bus (MIL-Std-1553B) towards Ground (POC) is substituted by an Ethernet/UDP link. The Robotic Control Bus (CAN) and Servo-Control-Board (SBC) are here emulated. Master node runs the major HiPeRCAR Software.

- ***Solid-State Mass Memory (MM)***

A second PPC board substitutes the Solid State Mass-Memory (Worker3 in the figure 2-1) with its own RAM. A simplified file-storage system allows to save periodically a fixed amount of data for reconstructing the context environment of the Worker nodes.

- ***Two Worker Nodes***

Two PPC405 boards provide the required high processing capability (greater than 300 MIPS) with acceptable power consumption (20 W).

- ***SpaceWire Terminal Interface***

A SpaceWire FPGA on each node provides a full mesh of 30Mbps point-to-point links. The SpW FPGA (ECSS-E50-12A compliant) has been developed just for the HiPeRCAR Project.
No need of router exists in this minimum system.

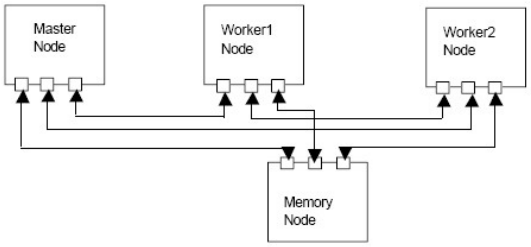

Fig. 2-2 – Logical Scheme of HiPeRCAR Target

Tests have been performed on HiPeRCAR Target, connected via UDP to POC which sends commands and receives telemetry. DREAMS program on POC computer allows the representation of data by means of a post-processing Matlab program.

3. RESILIENCE IN HiPeRCAR

The concept: "Resilience" permits a system to react to possible damages by reallocating the faulty tasks in optimal mode.
HiPeRCAR system implements by a personal design this concept.

Basic assumptions:
a) The wanted power is given by commercial-grade computers.

b) Failures (due to cosmic radiations) apply to the weakest nodes, i.e. the Worker Nodes based on commercial computers. They produce temporary of definitively degradation of the affected node. Rad-hard nodes (Master and MM) are failure-protected.

Objective: HiPeRCAR guarantees that, despite of failures of the Worker nodes, the system provides continuity of the services and resumes the original performance in minimum time.

HiPeRCAR system tolerates a reduction of the performance but not stopping of service. It has to autonomously resume the nominal performance as soon the failure effect ends. If the failed node cannot be recovered, the system will assign the affected jobs to other available nodes or in worst-case the Master will carry out by those jobs with the available resources.

3.1 Resilience Strategy

In case of node failure, shutdown and reboot take a long time unacceptable for the continuity of the service. Therefore, a safe control relays on tasks which

run permanently on the rad-hard Master that never stops.
In order to get this objective, the HiPeRCAR Software maintains a double version of the robotic tasks running simultaneously on Master and Worker Nodes.

Worker nodes run the full operational version, the *Nominal* one, while Master node carries out a reduced, or *Basic*, version of the same task.
Master provides by alone a minimum control of the robotic devices in any case; so, if the result of the Nominal task is missing, Master guarantees autonomy and reliability.
In fact, if a Worker fails just its contribution goes lost, i.e. the advanced commands are missing, but the basic controller stays operable. An external observer does not detect discontinuity in the service.

3.2 Resilience Design

The whole process is managed by the Master Node. Master performs a continuous monitoring on Workers: the progress of their tasks is under control (Fault Detection**)**. The robotic task is made by the cooperation of Master and Worker (Nominal Mode), but the presence of the latter is not an absolute need. When the Worker contribution is missing, the system goes instantaneously from Nominal Mode to Basic Mode (*robustness action*).

In Basic mode, Master carries out a simplified version of the robotic task implemented nominally by the Worker avoiding the interruption of the service. In the meantime, Master tries to reboot the faulty Worker. If this does not help, the faulty Worker is turned off definitely and a spare Worker is activated, if available.

Ended the node fault recovery, the Configuration agent reallocates the advanced robotic tasks on the Worker returned to its operational status (*resilience action*). If this fails, the system stays in the basic mode (*graceful-degradation action*).
In any case, there is no interruption of service.

3.3 Control-Task Design

Each robotic service is a control task devised as made of two internal stages: "*Nominal*" and "*Basic*" mode.

Nominal Mode foresees complex and resource-intensive computations provided by powerful Worker nodes and network. The outputs of the Nominal tasks are achieved by the cooperation of Worker and Master Nodes.

Basic Mode performs only safe operations. Its design is simple, resources economical, in charge to Master Node. While running in basic mode, the system provides service at low performance.

Once the above tools are available in HiPeRCAR system, splitting a given robotic function into Basic and Nominal mode is the major challenge of the Project. There are no general rules to split a robotic task into complementary parties; this design activity is done on a case-by-case basis: PASTEUR and EUROBOT missions provide a suitable scenario to carry out this exercise.

The robotic control tasks execute on the basis of a time-slice, named 'tick'. This is the task SW-cycle or the atomic unit on which the task must be completed. Faster is the CPU, shorter can be the tick unit. Here, the robotic application uses a 20 ms tick as suitable SW-cycle for this kind of applications.

The HiPeRCAR system operates by means of the efficient and transparent services provided by the Middleware Communication Manager, in charge of distributing messages among tasks everywhere located over the network. The inter-nodes communications is supported by a high-speed SpaceWire network.

Middleware runs on BOSS, the real-time kernel [4], which makes a virtual hardware abstraction allocating SW tasks everywhere on the system net.

Workers periodically save their context environment into a dedicated area of the Mass Memory. Storing helps to recover an interrupted status; in fact, Worker retrieves the last status from MM at node reboot or task moving.

4. PASTEUR TEST SCENARIO

'*Pasteur*', the driller system of *Exomars* mission is a suitable study-case with its operative scenario. The Pasteur test-case has been took from the Test Plan [6].

The Mission Application runs on top of the Middleware Framework which implements al the basic (mission-independent) services (like 1553-communications to/from POC, TC&TM, Operative Modes, CAN robotic Bus, FDIR).

The Pasteur application is envisaged as made of four control tasks: (1) Drill Control Task, in charge the handling of drill mechanisms; (2) SPDS&Microscope Control Task for handling the Sample Preparation and Distribution System and Microscope; (3) Data Acquisition Control Task, for handling the acquisition and storing of mission data; (4) Mission Autonomy task, for handling mission plan.

Each control-task is designed by three main components: the Basic task running on Master, the Nominal task running on Worker, the Nominal Context Descriptor handled by MM.

The kind of implementation has its fundamental on some basic concepts:

1. Basic and Nominal related algorithms must have the same duration.
2. Nominal control-tasks have no-memory of the previous activities. In case of failure, Nominal task can not know if/when/how-long it was not active. So, Nominal activity shall not depend on previous activities.
3. Nominal control-tasks execute without performing consistency check. Nominal can not perform checks on any part, either HW or SW.
4. Context information should not contain status information, because Nominal can not use it.

The paper considers one case only for sake of clarity. The Drill-Control-Task is examined just to catch the HiPeRCAR features.

5. DRILL-CONTROL DESIGN

Drill-Control is a task in charge of handling the drill mechanisms.

Three main algorithms have been implemented: Position-Control for translation mechanism; Speed-Control for drill rotation mechanism; Controlled-Stop for drill rotation mechanism.

Nominal and Basic tasks run different Position-Control and Speed-Control algorithms (see Fig.5-1), being the Nominal more accurate than the Basic ones; a complex polynomial expression in Nominal is simplified by a linear expression in Basic Mode. The Controlled-Stop algorithm, since very simple, uses the same linear expression for both Basic and Nominal Modes.

Relations between Basic and Nominal are achieved by an intensive communication based on standardised messages.

Two kinds of messages are exchanged between Basic and Nominal tasks (ExecuteTrajectoryPlanning and ExecuteTrajectoryStep). Maser starts the communication with a message with and waits for the relevant reply from Nominal.
Deviations from this protocol activates the Basic generation of setpoints to be issued on the robotic control bus.
Relationship Basic/Nominal requires an intensive communications because it is necessary to refresh at each SW-cycle the actual position of drill mechanisms since "*no assumptions have to be done on the previous status of Nominal*".
The message exchange is supported by the high-throughput SpaceWire network.

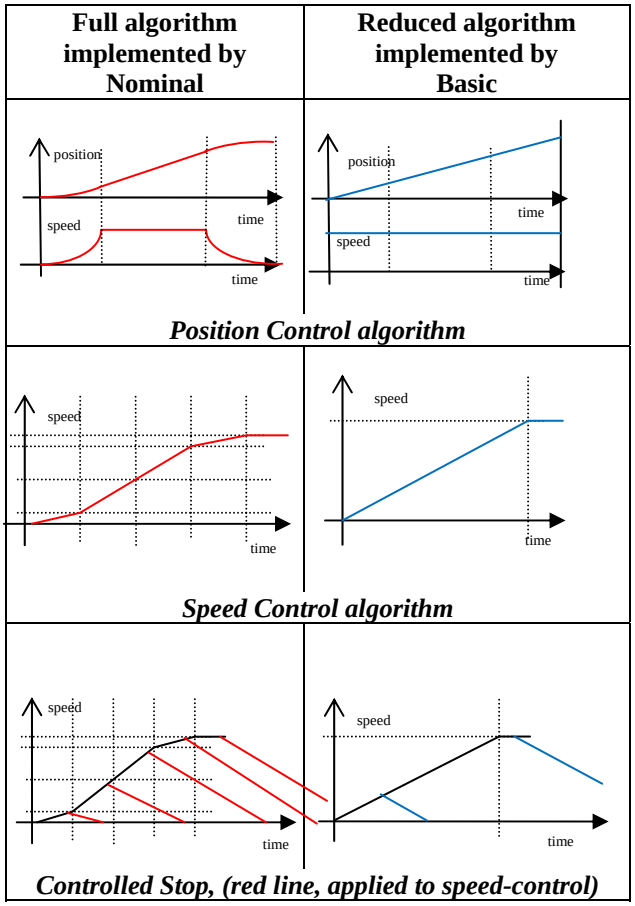

Fig.5.1 – Drill Control Tasks

The design of Basic Mode is based on four steps:

Start_infinite_loop
step1: check if a new drill-control command is received by POC. If so:
- check if the command is allowed wrt the current drill status
- check the correctness of the command
- prepare the message ExecuteTrajectoryPlanning towards Nominal

step2: Send message ExecuteTrajectoryPlanning or ExecuteTrajectoryStep to Nominal then evaluate the setpoint to be used as back-up solution if Nominal does not reply.
step3: Wait for next cycle clock time (tick)
step4: if a message has been sent to Nominal, check if a message-reply is arrived from Nominal and send the setpoints to external drivers.
End_infinite_loop

After sending the message to Nominal (step2) and before checking if the answer is arrived (step4), Basic is in step3 waiting for next cycle. During this waiting time, Nominal shall plan the new trajectory or evaluates next setpoint. Nominal shall plan the new trajectory within 20 ticks, and shall provide the setpoint in 1 tick.

Resilience in "Drill Control Task"
When a failure happens, according to step failed, Pasteur system reacts in this way:
- if the Nominal has a failure while Basic sends the ExecuteTrajectoryPlanning message (step2): the Basic sends the ExecuteTrajectoryPlanning for as a maximum 20 times, one message every tick; if no answer is received, the Basic will execute the trajectory planned by itself and does not contact Nominal for the whole duration of actual trajectory.
- if the Nominal has a failure while a movement is in progress (step3): the Basic, every tick, sends the ExecuteTrajectoryStep; if no answer is received within 1 tick, the Basic will send to the drivers the setpoints evaluated by itself (the backup setpoints). In the next ticks, Basic will send again to Nominal the ExecuteTrajectoryStep with the actual status: if answer is received, it will send to the drivers the setpoint provided by Nominal; otherwise, Basic will use the setpoints generated by it.

Once the Nominal is activated, again the TaskManager sends to Nominal the context. By using context and the actual status information sent by Basic, Nominal can properly continue its activity.

The Mission under test plans issuing by POC of the following commands:
- start acquisition
- drive on drill
- drill rotation 50 [rpm], acceleration time 2[s]
- controlled stop
- drive off drill
- stop acquisition
- download file DrillRotB

The off-line post-processor translates the generated telemetry data contained in the file DrillRotB into Matlab format to have an intuitive representation of the system behaviour.

6. TEST DESIGN

Firstly a reference test-case is generated by executing the mission plan without inducing failures. The Fig. 6-1 shows how *Nominal* Mode will perform for an undisturbed implementation of Speed-Control and the Controlled-Stop algorithms.

The outputs of the system are just the ones produced by *Normal* Mode running on Worker Node.

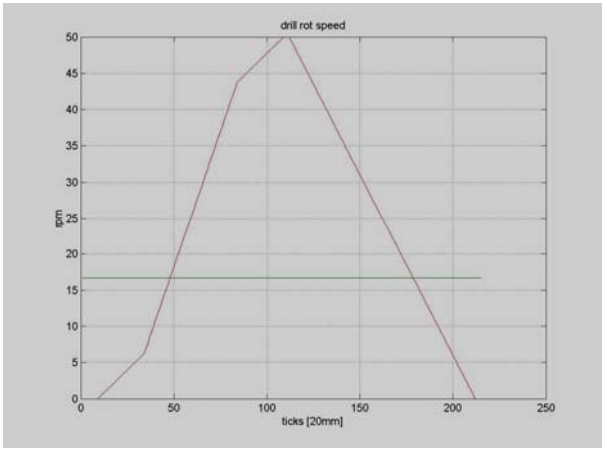

Fig 6-1: Drill rotational speed evaluated by Nominal

A second reference test-case for the same tasks is executed by *Basic* Mode only, that is de-activating the *Nominal* Drill Control and without failures induced. By this way, the outputs of the system are just the ones produced by *Basic* Mode running on Master Node.

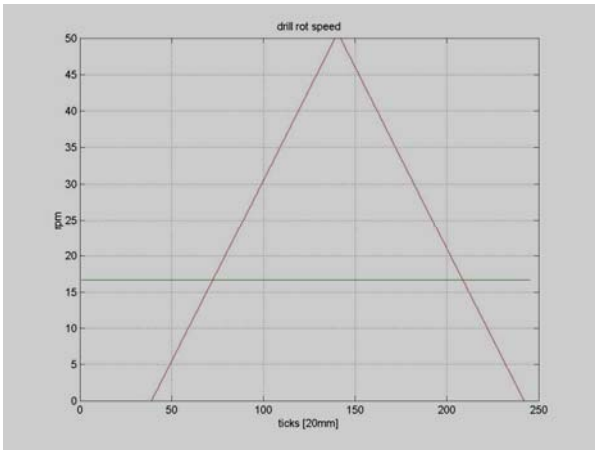

Fig 6-2: Drill rotational speed as evaluated by Basic

Fig. 6-2 points out the simplified behaviour achieved by *Basic* Mode.
Some previous tests allowed to get some measurements as the latency of task switching: FDIR spends around 20 ticks to confirm that Worker is gone out and 16 tick to reactivate a task on another active node.

Finally, we have in our hands the means to evaluate the system behaviour.
The test starts up all the nodes: Master, Worker1, Worker2 and MM. During the trajectory execution, Worker1 (blue line) and then Worker2 (green line) are both powered on.
The Drill-rot-speed is assigned to Worker2.

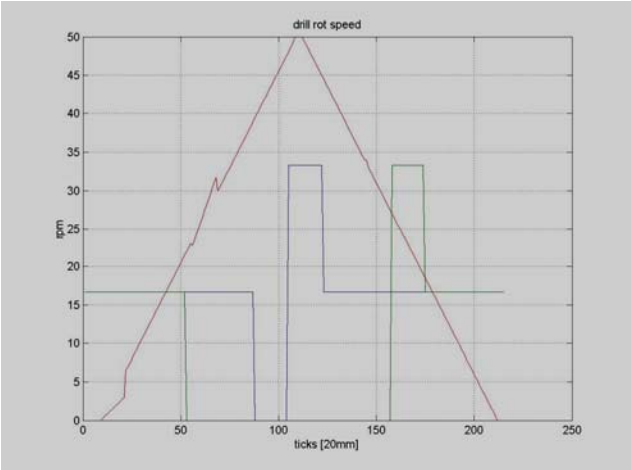

Fig 6-3: Drill rotational speed with injected failure

6.1 Results Analysis

The Fig. 6-3 allow to point out:
At start, both Workers are up.
Worker2 starts executing the task as per Fig 6.1. At tick 21, Worker2 goes in failure; Basic set-points are immediately provided (*robustness action*). The discontinuity on the plot is due to the application of Basic profile (see fig 6.2).

At tick 51, the system declares Worker2 off. The FDIR task spends 24 ticks to detect that Worker2 is gone off. 2 ticks are required to migrate the task onto the active Worker1; (some tests show that 20 ticks for FDIR and 1 tick for task migration are enough).
The latency of the system can be assumed of 20-25 ticks. But over this time, system stays always available.

From now on, Master performs the drill-rot-speed task through Worker1 (*resilience action*). The plot shows at tick 53 the discontinuity due to this switch.

Worker1 goes in troubles at tick 68 and its Nominal task does not respond: the system takes about 20 ticks to detect that a node is off; in fact, its shut down is detected at tick 88. Again, Basic intervenes immediately providing its own computed setpoints. The slope of the plot shows the use of the Basic curve of fig 6.2.

Deceleration phase: on this phase there is no discontinuity in the setpoint generated, because in the deceleration phase the same linear controlled-stop algorithm is implemented in Basic and Nominal.

Worker2 becomes active at tick 175, and Drill Control task migrates back to the default node (i.e. Worker2); no discontinuity is generated.

This test case does not show the graceful degradation, as never both the Workers are off.

7. CONCLUSIVE CONSIDERATIONS

The validation phase confirms the achievement of the main objectives of the HiPeRCAR Project.
HiPeRCAR, by means of all its components Hardware and Software (see Fig. 1), shows that the Basic/Nominal strategy guarantees *availability* and *resilience* able to face failures on the weak part.

In particular, HiPeRCAR shows the following features:
Availability: no-stop of the robotic tasks thanks to the prompt application of the reduced service version. When Nominal does not give its contribution, Basic provides by alone the outputs: no-loosing of setpoints towards drivers.
The '*robustness action*' avoids the out of the service. The reduction of performance achieved in this test-case is due to the simplification of Basic mode; a more sophisticated implementation of the Basic algorithms could help by smoothing the discontinuity points.

Readiness: The transition Nominal-to-Basic is immediate, since Basic is designed to compute in advance its outputs before receiving the Nominal outputs.
The transition Basic-to-Nominal requires 1 or 2 ticks (20 ms in our case) when another node is available.

Resilience: When a failure is detected, the system restores the service in 1 or 2 ticks on another active node.
When possible, the system spends about 20 ticks to re-start (power-off/power-on) the service on the same node. A 50-Hz system restores the full capability in 400 ms.
If the resilient action is not successful, because of the permanence of the faulty status, the system continues the job in degraded mode ('*graceful degradation action*').

The validation test campaign provides the results of several test-cases, which allows to get a fine picture of the system performance and of the algorithm design [6].

But, the test campaign highlighted also some topics that could request to continue the study on the future:
- Task downgrading from Nominal to Basic is a very delicate design, subject of optimisation;
- More sophisticated Basic algorithms lead to reduce discontinuity and phase-inversion in the control behaviour ('negative' speeds are not admitted by a real Controller);

- SSMM: saving/restoring rich context-environments greatly increases the current '*stateless*' design of Worker nodes. Missing information on last status overloads Master Node which has to update continuously workers with their last status
- Powerful computer boards allows overloading of Worker nodes that can maintain more Control Tasks in dormant-state
- Powerful Master computer allows to manage more Worker Nodes by running their Basic modes.
- Master Node is charged of too many tasks in the current design; FDIR can take great advantage by distributing outside some tasks; e.g. the Robotic Bus Controller can be allocated to a real CAN-controller (SBC) like the 'RTI' (available ESA-project).
- Master Node implementation based on Leon Processor.
- The system architecture can be extended by introducing a SpaceWire router to support more Nodes in the loop.

These points and other ones could be study-cases for the continuation of the HiPeRCAR project.

REFERENCES

[1] D. Jameux, "Application of the Payload Data Processing and Storage System to MOSREM (Multi-processor On-board System for Robotic Exploration Missions)" - proceedings of DASIA 2003

[2] Guy Estaves, Alcatel Alenia Space – "SuperComputers for Space Applications"- SDSS 2005, 17-20 October 2005, ESTEC, Noordwijk

[3] P.A. Marra, S. Montenegro, E. Crudo, F. Fusco, D. Akautse, R. Vitulli – "HiPeRCAR: the High Performance Resilient Computer for Autonomous Robotics" - DASIA 2006, 22-25 May – Berlin

[4] R. Vitulli, S. Montenegro, "High Performance Ultra High Dependable Architecture for Autonomous Robotics in Space" - ASTRA 2006, 17-20 October 2005, ESTEC, Noordwijk.

[5] P.A. Marra, S. Montenegro, E. Crudo, F. Fusco, D. Akautse, R. Vitulli – "Assessment of the Resilience Concept by means of the HiPeRCAR Simulator" - DASIA 2007 29th May - 1st June 2007 Naples, Italy

[6] "HIPERCAR – Target Demonstrator Test Report"- HI-AASIM-TR-0002 - Thales Alenia Space Italia - (Milan)