LVCUGEN (TSP-based solution) and first porting feedback

Julien GALIZZI, Jean-Jacques METGE, Paul ARBERET, Eric MORAND, Fabien VIGEANT [(1)], Alfons CRESPO, Miguel MASMANO, Javier CORONEL PARADA, Ismael RIPOLL, Vicent BROCAL [(2)], Florence ROUBERT, Christophe SCURI, Vincent TEDESCO [(3)], Nicolas THOMASSON [(4)]

[(1)]Centre National d'Etudes Spatiales (C.N.E.S.), 18 av. Edouard Belin - 31401 Toulouse CEDEX 9 France – <name>.<surname>@cnes.fr

[(2)]Instituto de Informatica Industrial, Universidad Polit´ecnica de Valencia (U.P.V.), Camino de vera s/n CP: 46022 Valencia, Spain –
acrespo@disca.upv.es
iripoll@disca.upv.es
mmasmano@ai2.upv.es
jacopa@ai2.upv.es
vbrocal@ai2.upv.es

[(3)]Communications & Systems (C&S), 3 Rue du Professeur Vellas Europarc 1 - 31300 Toulouse FRANCE - <name>.<surname>@c-s.fr

[(4)]Sogeti HT, 485 avenue de l'Europe – 38330 Montbonnot Saint Martin France – nicolas.thomasson@sogeti.com

KEYWORDS : Time and Space Partitioning, IMA, Space, Hypervisor, XtratuM, ARINC653.

1- INTRODUCTION

In a context of complexification of embedded systems within satellites, CNES is an actor in the will to promote standardization and reuse from a mission to another.

By definition, in the frame of satellites payloads, the flight software is mission specific and therefore developed from scratch for each instrument. This is due to the fact that most of the payload developments in the scope of scientific missions are contracted to national institutes (each time a different one), those have generally poor background in embedded real time software engineering. But even in this case, a certain number of subsets can be considered as generic and would benefit in not being redeveloped for each mission.

In the frame of payload software, CNES has initiated studies and architectural activities [1] related to the use of TSP-based technology (Time and Space Partitioning). This solution, called LVCUGEN, consists in the virtualization of the hardware resources and in the isolation in time and memory of the different software subsets. Some are generic (and configurable statically according to the mission needs) and the others are mission specific. This will allow the scientific institutes to only develop the mission specific data processing and benefit from the reliability and availability of the other generic subsets.

The other side of genericity for the Flight Software is the capability to plug on different boards. In the actual European spatial missions, there is no standardized hardware board and each scientific institute may have the will to use a given specific board. That means that a generic flight software shall be able to be hosted by any board, with the less necessary adaptations to cover different processors, memories and also I/O specificities.

In the frame of the firsts developments of the payload computer for the SVOM project (scientific mission devoted to gamma burst detection), CNES has decided to experiment the LVCUGEN in a real context, with mission constraints and expected performances, and with its dedicated board : the CPU ITAR-FREE, based on AT697F (LEON2) and ATF280 (ATMEL FPGA) chips.

2- SVOM PROJECT

SVOM is a Sino-french satellite project dedicated to the detection, localization and study of Gamma Ray Bursts and other high-energy transient phenomena.
Its avionics architecture consists in a platform, 2 payload computers (a French one and a Chinese one), and 4 instruments developed by scientific institutes (from China, France, Germany and England).
The satellite is supposed to be launched in 2015.

The use case for the porting of LVCUGEN if the French payload computer, that must cover the following needs :

- send alerts in less than 2 seconds to the ground stations when a GRB is detected in order ground antennas to observe also the phenomena
- monitoring and control of the payload and the instruments
- ensure communication with the platform and the instruments
- take into account telecommands according to a given bandwidth and with a given reactivity
- route science data to the platform or to the ground through a dedicated antenna according to a given bandwidth and with a given reactivity

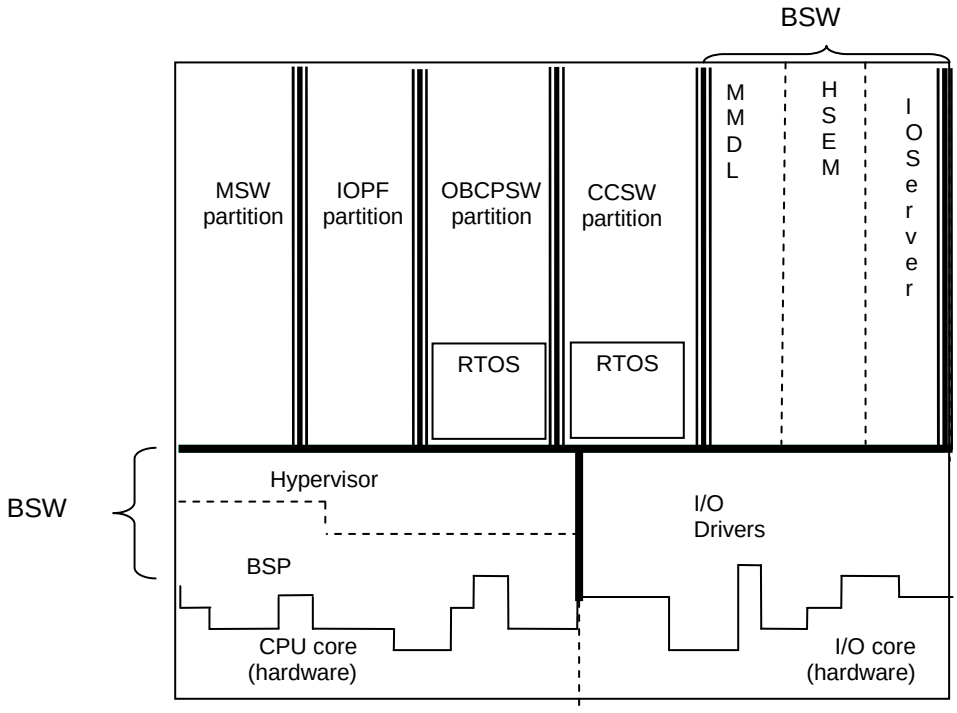
As a consequence, within this payload computer, from a software point of view, we can find processings specific to the HW, processings that are common to all the missions, and processings specific to SVOM specific needs.

3- LVCUGEN ARCHITECTURE

LVCUGEN is a TSP solution that decouples the TSP features from traditional RTOS.
It aims at providing to the space community, and maintaining an off-the-shelf framework, based upon a set of generic software building blocks, a qualified development process and associated offline tools.

It is based on a Basic SoftWare that contains :

- an Xtratum hypervisor (developed by UPV [2]) that manages the memory and time segregation between the different partitions (ARINC653 cyclic scheduling policy). Xtratum provides an API that allows, among others, the partitions to communicate through RAM ports with each other. Each of these partitions, according to its needs, can use a specific RTOS if the Xtratum API is not sufficient (i.e. if multithreading is required).
- I/O drivers libraries that offer standardized interfaces that can be used by the partitions that need to access directly to I/O devices.
- a partition MMDL that manages the software modes, software patches, software configuration control, and memory dumps at computer level.
- a partition HSEM that manages the payload computer Failure Detection, Isolation and Recovery inside the computer.
- a partition IOServer that manages the shared I/O inside the computer.



In the frame of LVCUGEN BSW, it was initially planned to develop as generic 2 partitions :

- one to manage the Mass Memory but the need was not confirmed for SVOM and it will be developed later, as the need is still present for other projects.
- one for the in flight instrumentation of the computer. It will be developed later.

One of the interests of TSP is to decouple the execution contexts of mission specific software from the one of the payload monitoring and control software.
In the frame of the standardization, a partition CCSW, has been developed using RTEMS as RTOS (that has been ported on Xtratum) with the objective of the genericity to cover the space services for monitoring and controlling the payload from the ground. At the moment, it only allows a strong reuse between projects but genericity is kept as objective for middle term.

This framework allows to host specific mission processings in one or more dedicated independent partitions.

The allocation of resources is made using a simple, robust and safe hypervisor configuration process, allowing to ensure the respect of partitioning :

- hypervisor based in the case of CPU and memory
- I/O server based in the case of shared I/Os.

4- THE LVCUGEN –INTEGRATION ON CPU ITAR-FREE BOARD

The CPU ITAR-Free board is composed by the LEON2 AT697F (without MMU, 32KB of instruction cache, 16 KB of write-through data cache) as microprocessor and the FPGA ATF280 from ATMEL. It supports the following I/Os : I2C, OSLink, Spacewire, Flash SPI, LICE and UART, accessible through a set of DMA controllers.
This board provides 1 MB of Flash SPI, 1 MB of SRAM directly connected to the internal AT697F memory controller and 128 MB of SDRAM accessible through the ATF280 chip. The SDRAM is shared between the µP and the DMA controllers.
AT697F runs at 80 MHz, while ATF280 runs at 20 MHz, that makes the accesses to the SDRAM very slow.
In order to manage the communication with the Platform through 1553, an OSLink-1553 gateway (ModBus board) has been provided.

We will only present here after the integration aspects that deal with partitioning and problematic linked to the integration of LVCUGEN onto the board.

Several problems (anticipated or not) emerged during the integration :

- No Memory Management Unit : on the version of LEON2 used, no MMU was available but only a Write Protection Register.
    - This register allowed us to allocate for writing only one area to each partition. It was sufficient for our case to prevent the areas of the partitions to be written by other partitions. But in opposite to classical Time and Space Partitioning environments, we do not guarantee that the areas of one partition is not readable by other partitions.
    - If a partition needs to write in others areas that the one that is allocated to her, like in I/O registers, then the integrator shall give the write access for specific addresses to this partition in the System configuration, ensure that this address is not allocated to other partitions, and then the considered partition will have the possibility to write at this address using a specific XtratuM API call.

- Determinism of bandwidth for shared I/Os : as the hardware did not allow to configure and manage the bandwidth for each media, and due to the problem raised by the concurrent DMA accesses, it has been chosen to provide a generic I/O server partition that manages all the exchanges for shared I/O and that communicates through RAM ports with the other partitions.

- Respect of timing constraints : each partition can have several timing constraints to handle : tasks requiring a given CPU time, a given periodicity, deadlines, precedence between tasks… It appeared quickly that, to respect all the timing constraints for all the partitions with very frequent slots of IOServer, a tool was necessary to generate a scheduling plan assuming all the constraints.

The tool Xoncrete, developed by UPV, has been “massively” used in order to enable us to build a scheduling plan allowing to meet all the periodicities and performance constraints. This tool needed to be upgraded in order to enable us to add hardware constraints in the scheduling plan : while requesting a DMA access to an I/O device, we can configure a time where no task accessing to the same DMA will be scheduled and then reduce constraints on the scheduling.

- Evolutions of the scheduling plan : one consequence of having a complex scheduling plans with 150 slots of various durations is that any change in the timing constraints will modify the slots allocated to each partition and thus their context of execution. As the commitment on performances by each partition is strongly impacted by the characteristics and the number of its slots, it is very important to guarantee that a partition will not be impacted by an evolution of the scheduling plan. From an industrial point of view, it was thus mandatory to add a feature to the Xoncrete tool in order it to allow the user to manage incremental scheduling plans : the user has the possibility to save a scheduling plan, to freeze the slots of some partitions, and to modify the constraints of other ones to reschedule them without impact on the frozen ones.

- Partitioned access to I/O driver or I/O device, or none ?
  When a partition needs to access to a device, it uses the drivers provided by the BSW. Using a driver means programming registers and in case of partitioning, if a given register is required to program all its devices, then we can not have 2 partitions using this driver. The case occurred on I2C where a register is used to initialize the 3 devices. The workaround was to perform the initialization of these 3 devices inside IOServer, even if 2 of them are used by FDIR.

- Access to 1553 interface : to communicate with the Platform using 1553 bus, a ModBus board is used. It contains a FPGA, an 1553 IP, some RAM memory and we can access and control it using OSLink. The problem linked to this design is that it complexifies the 1553 interface management, introduces additional latency and put significant temporal constraints (coming essentially from 1553 high level protocol) on software processing, which can lead finally to CPU overhead in case of LVCUGEN architecture.
  Moreover, this solution does not allow to be generic at software level. With the others I/O types (spacewire, I2C), we can develop drivers that provide standardized interfaces for initialization, read, write, … but in this case, several OS Link accesses are required to perform a single write operation to emit a TM. In this difficult context, it has been necessary to manage this interface as a private I/O in a dedicated non-generic partition that manages the exchanges with the platform.

- Synchronization with the Platform : on satellites, all the computers are synchronized on an unique On-Board Time. The tick of a new second is propagated through a discrete signal that provides an high resolution to the On-Board Time and to the events. In our case where the time is allocated statically within one fixed period, it has been necessary to extend Xtratum capabilities, in order to implement an external synchronization capability. The consequence is that the period of the MAjor Frame (MAF) is set to 1 second and that every new MAF is launched upon reception of the discrete signal representing the On-Board Time.

- Impossibility to use the cache snooping capability of the LEON : in order to increase performances, data and instruction caches of the LEON have been massively used. The design of the board did not allow to use the cache snooping capability and it is necessary to flush the data cache before any access to DMA RX buffers (performed only in case of high speed synchronous access because TSP already flushed the cache at each partition switch).

- Management of synchronous communications : in this experimentation, regarding 1553, we took soft hypothesis with only an external synchronization (period 1s). We could face hard problems if a strong reactivity is required that can not be managed by the hardware by itself. Because assuming a strong reactivity leads us to strong scheduling constraints that can become very consuming in CPU.

5- FUNCTIONAL TESTS AND PERFORMANCES

The partitions related to Monitoring and Control management, communication with the platform, and scientific processing were not developed with full functionalities but can be considered as sufficiently representative of a real flight software.

The tests showed that the architecture can support the SVOM worst cases.

They were tested with the worst functional cases in term of I/O :
- 1553 : 23 KBytes/s with 3 accesses every 125 ms
- I2C : 5 Kbytes/s with 9 operations every second
- Spacewire : 60 Kbytes/s with 1 acquisition every 25 ms and 1 emission every 50 ms.

The FLASH and RAM memories show the following consumptions :

Component	**FLASH**	**RAM**
XtratuM (including channels)	85 KB	934 KB
XtratuM Configuration Tables	36 KB	36 KB
IOServer	57 KB	91 KB
Modes Management and DataLoad	52 KB	1,1 MB (to upload up to a 1MB binary)
Failure Detection Isolation and Recovery	57 KB	75 KB
IOPlatform	49 KB	73 KB
Monitoring & Control partition	267 KB	800 KB
OBCPSW partition	252 KB	800 KB
Mission partition	38 KB	104 KB

It is important to detail that the communication channels between all the partitions (that is directly linked to specific SVOM needs) is included in the XtratuM RAM and represents 670 KB.

The CPU allocation showed that :
- the generic aspects (BSW) cover 23% of the CPU, each partition having a strong mastership of its worst case execution time,
- 31% are allocated to PUS management (all used due to the use of aperiodic background tasks),
- 14% are allocated to the communication with the platform (max. used : 5%),
- 10% are allocated to the java OBCP processing (all used because it is a background task),
- 20% are available for the mission processing (max. used : 2%).

All these tests have been performed with :
- cache enabled at any level (Xtratum and partitions)
- cache flushed between each context switch (TSP constraints)

We can also present the Virtualization and TSP overheads, in SVOM worst cases, compared to a classic monolithic architecture.
- virtualization overhead is computed adding :
  - overhead linked to memcopies (messages exchanges) between BSW and the other partitions,
  - nb_slots_BSW * (last_instant_hypercall + partition context switch).
- TSP overhead is computed adding :
  - overhead linked to memcopies (messages exchanges) between non-BSW partitions,
  - nb_slots_non-BSW_partitions * (last_instant_hypercall + partition context switch).

This lead us to the following results :

TSP overhead	2.7%
Virtualization overhead	2.5%

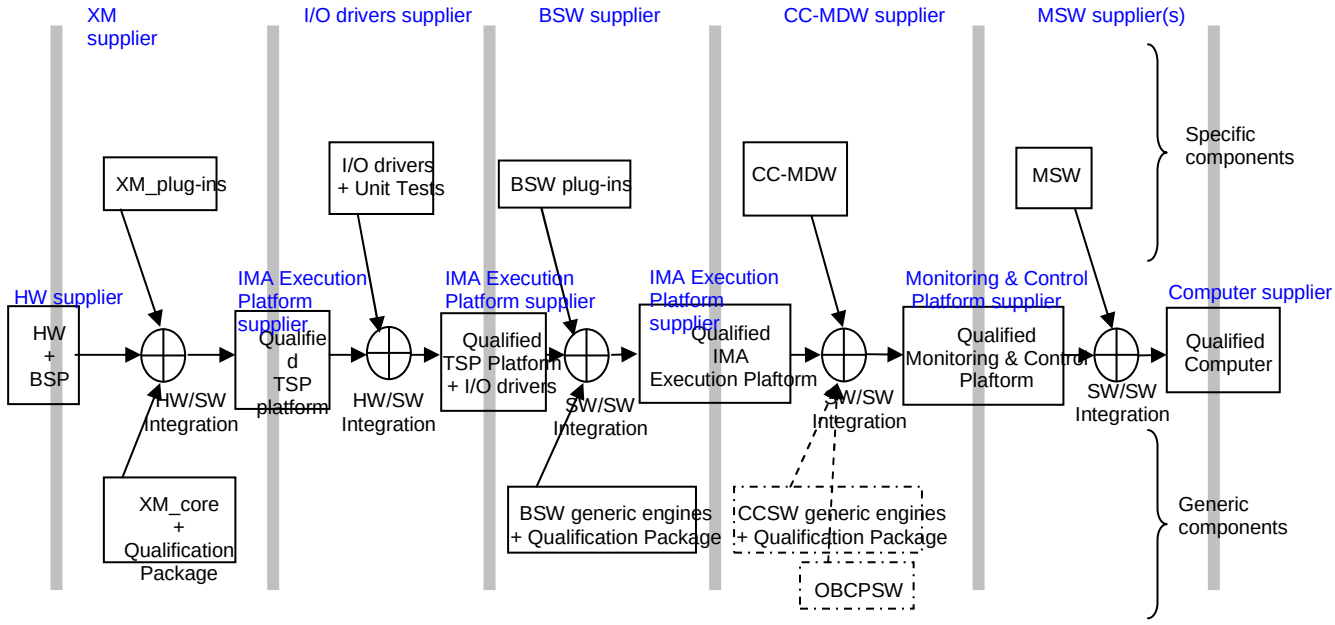
TSP + Virtualization overhead	Between 5 and 5.5 %

6- PROCESS, ORGANIZATION AND TOOLS

One of the main feedbacks of this porting is the necessity to provide to the users, at any level, a precise process to guide them to define and build an integrated computer, starting from a bare Hardware.
In this environment, the role(s) of Integrator(s) is a central one. He will ensure that any of the building blocks has been correctly integrated with the others, taking all the constraints into account.
To help him in his work in an industrialization process, some tools will have to be provided :

- Xoncrete, already presented in §3.
- An automatic verification tool to verify that the configurations produced respect all the constraints at any level (HW, XtratuM, System).
- A tool allowing any user to specify into library rules the envelope context where its building block is qualified. These library rules will be used as input by the automatic verification tool described above.
- A tool allowing to describe all the interfaces between all the partitions.
- An automatic code generator in charge of reading/writing/formatting/deformatting the messages on the basis of the description of the interfaces.
- Qualification kits allowing to qualify any of the generic building blocks on the target, after integration with their associated plug-ins.

The following figure describes the process as it should be when the industrialization phase will be launched.



All along this process (and even during the operational phase), the integrator can modify the allocation of resources and/or the constraints of any of the building blocks to reach the System needs.

7- STATUS – RESULTS – SCHEDULE

With this experimentation, we faced several problems linked to the use of a board not designed for high performances and partitioning (principally linked to the management of I/O) but we succeeded in solving these issues, developing the appropriate tool, or adapting our design to take into account Hardware specificities. These adaptations were mainly located in I/O drivers of the Basic Software and in the client partitions of the Basic Software, which gives a good confidence in the fact that the definition of the generic parts of the Basic Software have reached an encouraging maturity point.

We demonstrated also that the LVCUGEN was a very performing solution in the frame of SVOM project, allowing the segregation of faults, the simplification of determinism for the on-board software, the limitation of side effects between components, and providing generic tools for all the user missions. Moreover, the framework has been designed to have the less memory and CPU footprint, and the result is very encouraging.

A generic component developed by TAS that manages OBCP (automatic PUS procedures in Java), has been hosted by a new partition using RTEMS and executes nominally on the board. The porting raised no problem and went quite fast (4 weeks).

Another RTOS (Lithos from UPV) compliant with ARINC653 API has been evaluated as RTOS for our CCSW partition.

With regards to its promises for scientific missions, other projects and scientific institutes are interested in using LVCUGEN (Avionics-X for the future launcher) and have started its evaluation (like IAS on the project EJSM to explore Jupiter).

This framework raised big interests in a cross domains context (spatial, automotive, aeronautics, …) as its applicability is not limited to the only spatial and payloads needs. We are currently trying to support the emergence of a cross-domain initiative to improve the maturity of the overall framework and improve its business model.

Such a framework will change our usual ways to work and will probably modify our organization inside projects. But it also opens new possibilities in industrial organizations of satellites projects, optimizing genericity, reuse, and mastership of interfaces, while maintaining independence between applications and suppliers. This represents the future to manage an always increasing complexity for processings inside critical computers.

REFERENCES

[1] P. Arberet, J.J. Metge (CNES), O. Gras (Sogeti HT), A. Crespo (UPV) – *DASIA 2009* – **TSP-based generic payload on-board software.**

[2] A. Crespo, I. Ripoll, M. Masmano (UPV), J.J. Metge (CNES) – *DASIA 2009* – **XtratuM : an open source hypervisor for TSP embedded systems in aerospace.**