

# **THALASSIM CONCEPTS AND PRINCIPLES OF MODELLING**

**10th International Workshop on Simulation for European Space Programmes  
7-9 October 2008  
at ESTEC, Noordwijk, the Netherlands**

Bratulic H.<sup>(1)</sup>, Doussaud D.<sup>(2)</sup>

<sup>(1)</sup> *Thales Alenia Space*

*Email: herve.bratulic@thalesaleniaspace.com*

<sup>(2)</sup> *Thales Alenia Space*

*Email: didier.doussaud@thalesaleniaspace.com*

## **INTRODUCTION**

Like any complex real-time system, a spacecraft is submitted to validation tests, in order to verify its proper behaviour (on-board software and hardware) under flight representative conditions. In order to avoid a late discovery of anomalies during the spacecraft Assembly, Integration, and Test (AIT) phase, a comprehensive sequence of validation is usually conducted on the flight software, in a real hardware environment. This hardware represents a heavy investment, with a costly maintenance. In the frame of the Spacebus 4000 Telecommunication Spacecraft product line, our objective was to improve the competitiveness for the Avionics Validation sequence, performed after the development phases.

## **GOAL OF THALASSIM**

The main objectives of the Thalassim project were:

- To develop an Avionic Spacecraft simulator equivalent to the avionic hardware test bench, capable to check the On Board SoftWare patches (ie. use of real OBSW binary) and to model avionics buses and satellite harness, while avoiding a costly investment in an additional test bench with real H/W (see Fig. 1).
- To provide an alternative solution in case of failure of an electrical model used in the avionic test facilities, thus avoiding a single point failure in the avionic validation process.
- To increase our test capacity, in accordance with the commercial sales.
- To cope with a big variety of Spacebus configurations.
- To reduce the Avionics validation cost.

In addition, Thalassim allows to guarantee the in-orbit support (after sales services) in case of in-flight anomalies during the 15 years lifetime of each spacecraft (which means having test bench during up to 25 years).

The Avionics Validation and Qualification logic is given in Fig. 1.

## Avionics Validation & Qualification Logic

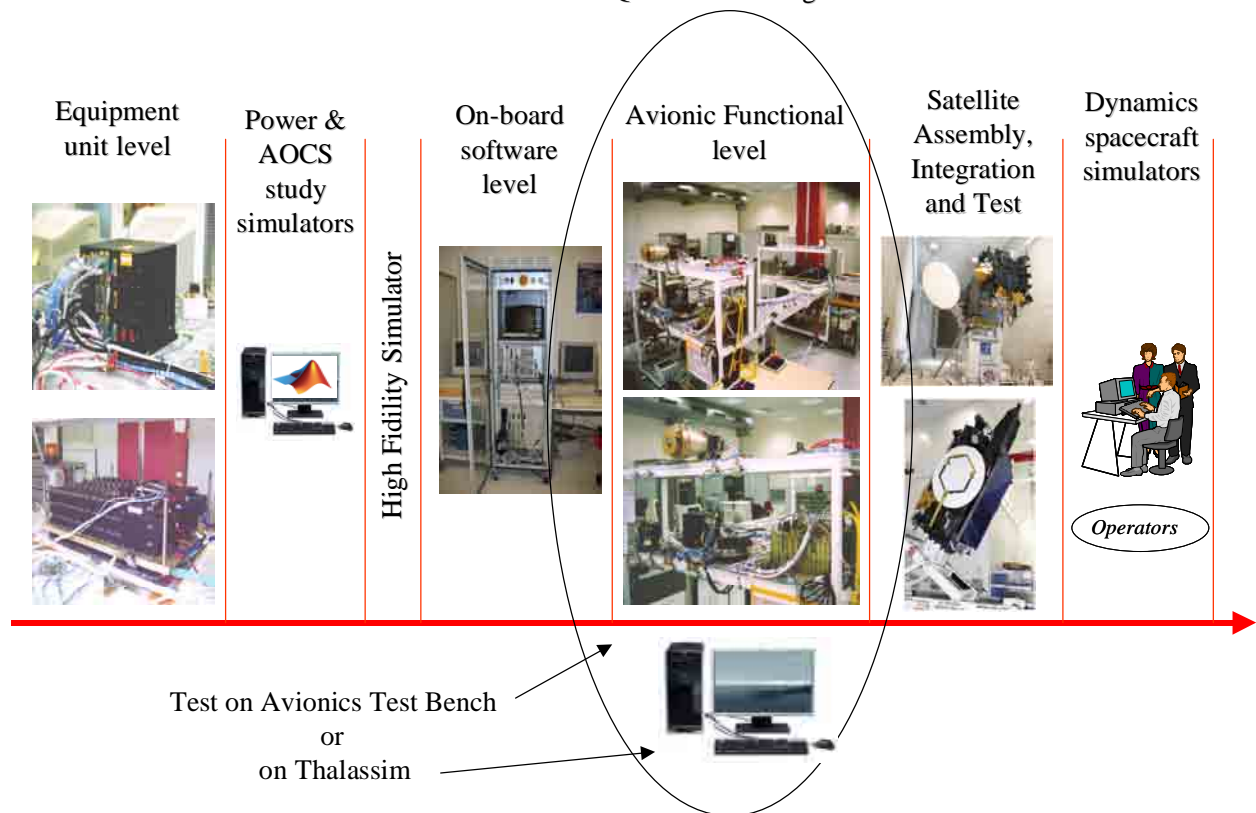


Fig. 1 : Avionics Validation & Qualification logic

Thalassim is a 100% software product designed and used as :

- Validation tool for the :
  - Performance and Robustness Verification with High Fidelity Simulator,
  - Onboard SW Verification Facility (SVF),
  - Avionic Functional Chains Validation (FCV),
  - Satellite Assembly, Integration and Test (AIT) procedures,
- A support tool for the :
  - Validation of Spacecraft operational procedures,
  - Validation of Satellite Control Centre,
  - Spacecraft operators,
  - Analyze and/or investigation of the anomalies detected in flight.

Thalassim covers the B, C, D and E project phases.

Thalassim is based on :

- New simulation platform developed by Thales Alenia Space for this project (called K2).
- New concepts : coupling and exploitation of the Satellite Data Base, automatic assembly of simulator, ...
- New principles of modelling : avionic 1553 and OBDH buses, system alarms, clock and synchronization interfaces, matrix command generator, digital matrix acquisition.
- An ERC32 simulator (TSIM of Gaisler Research Company, the only non open-source external tool of Thalassim).

Its architecture is modular and object-oriented.

### CONCEPTS

Thalassim is based on K2 infrastructure which integrates a set of complementary components:

- Infrastructure kernel (SMP2 compatible types and time keeper, scheduler, etc.).
- K2 models : This is a library containing all the models useful in the simulator instance. At each K2 model corresponds a dynamic shared library.
- Satellite Data Base Extractor: set of tools used to fill simulator data base from the THALES Satellite Data Base. In particular, the extracted data identifies the models and their interfaces according to equipment data sheets. The links between these interfaces are extracted from satellite harness definition.
- Simulator data base: contains the description of simulators, models, instances, interfaces and the functional links between instances.
- "Code generator": this tool is used to produce the skeleton source code of the models to be developed.

- “Simulator generator”: The simulator is generated automatically using the simulator data base.
- Test environment: environment which instantiates model and stimulates it from Python scripting language. Non regression tests are performed using this facility.
- Simulator control system: two ways are provided, standalone execution or remote control execution via the network (XML-RPC) from any client. Developed remote clients are : JAVA graphical monitoring and control, and command line Python tools. C++ remote client development is in progress.

The simulator concepts and design can be reused in terms of :

- Breakdown of the spacecraft into small building blocks reuse to different project phases.
- Extraction from Satellite Data Base (Equipment IDS and S/C harness).
- Architecture of the bus simulation (OBDH and 1553). Bus event link (dynamic routing of event using data contains in event) allows to connect all remote terminals to a single Bus controller.

Figure 2 hereafter illustrates us the link between the satellite Data Base, the simulator Data Base, the models and the instance simulator.

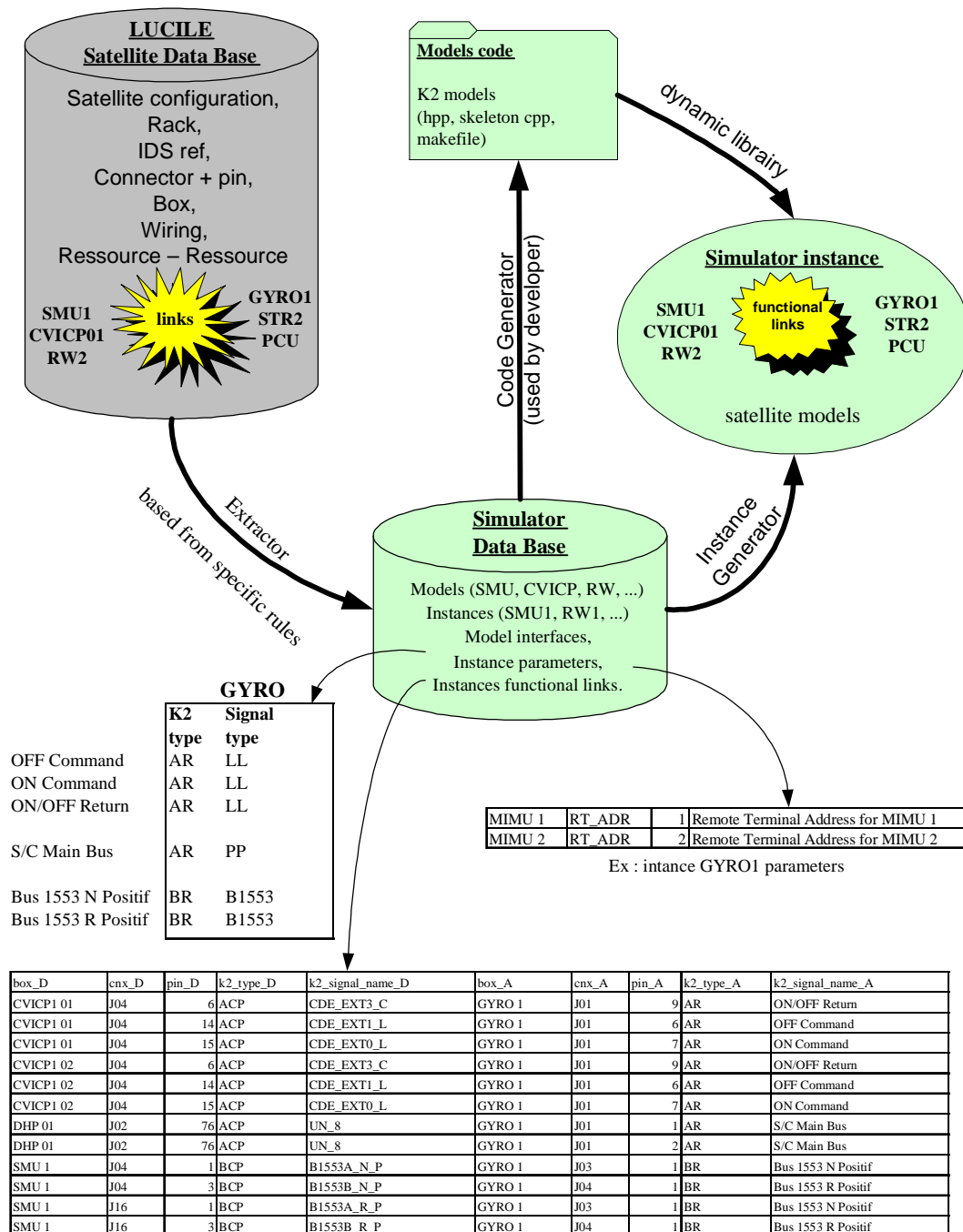


Fig. 2 : Links between the satellite Data Base, the simulator Data Base, the models and the instance simulator.

## K2 MODELS

This is a library containing all the models useful in the simulator instance.

A K2 Model is a software module (dynamic library) whose goal is to reproduce, as much as possible, an equipment or an environment. It must respect an interface provided by K2, containing all methods to :

- Initialize and finalize the model.
- Save and restore the context (states, internals).
- Provide access to the model from the K2 platform.
- Implement algorithms to simulate the model behaviour.

The K2 Models are generated from the Simulator DB by the Code generator component, and then compiled and linked within the simulator instance. The model interfaces have a high level of representativeness (same interfaces than real equipment unit).

## MODEL CODE AND SIMULATOR INSTANCE GENERATORS

The Model code generator is used by the developer to produce the skeleton source code of the models to be implemented. For each model :

- A C++ header file (with a “.hpp” extension) : a full file that can be used as is.
- A C++ internal header file (with a “.hpp” extension) : a skeleton file that will be completed to add some customizations.
- A C++ body file (with a “.cpp” extension) : a skeleton file that will be completed to implement the relevant algorithms.
- A SWIG interface file (with a “.i” extension) : a pre-defined file that can be completed (to add some includes for example).
- A makefile : to compile the model.
- Skeleton of unit test : Python script.

The Instance generator is used by the integrator to produce the Simulator instance description in XML format with contents :

- Models interfaces,
- Instances creation,
- Overloads and wirings declaration,
- Models default states and features,
- Overload states and features,
- Algorithms scheduling table.

## K2 PLATFORM SPECIFICITIES

We have chosen to develop the K2 proprietary platform for Thalassim. This platform is very simple and exactly adapted to our requirements:

- The platform is developed in C++ and Python interfaced with Swig open-source tools (Python coded parts are only used during initialisation phase, and not during critical operation).
- A specific connection type “bus activation” allows simulation of bus connection (necessary for generation of connection using real harness description).
- Models may be coded in C++, C, Ada, Matlab. Extension to other language like Fortran is possible.
- The model description is extracted from C/C++ header file (most of our models are coded in C++ and do not need extra description). Specific bridges allow the direct usage of Matlab models (using Matlab RealTime Workshop).
- The development is compatible with SMP2 types and TimeKeeper to permit future development of bridge to/from SMP2 simulators/models (scheduled during 2009).

K2 platform has multiple facilities to create a simulator scheduler, logger, tracer, access script language etc. The paragraphs hereafter illustrate the mains types of connection.

To implement all equipment units connections in a satellite harness, the platform offers 3 types of connection.

### Input to Output connections

The output is computed by the original model.

The Input is like a pointer to the output. The Input is used directly in the algorithm of the destination model. The algorithms of the destination model can test that its respective input is connected. For algorithm efficiency, the test may be done at simulation initialization by the platform; the model algorithms realize direct access to pointed address (see Fig. 3 hereafter).

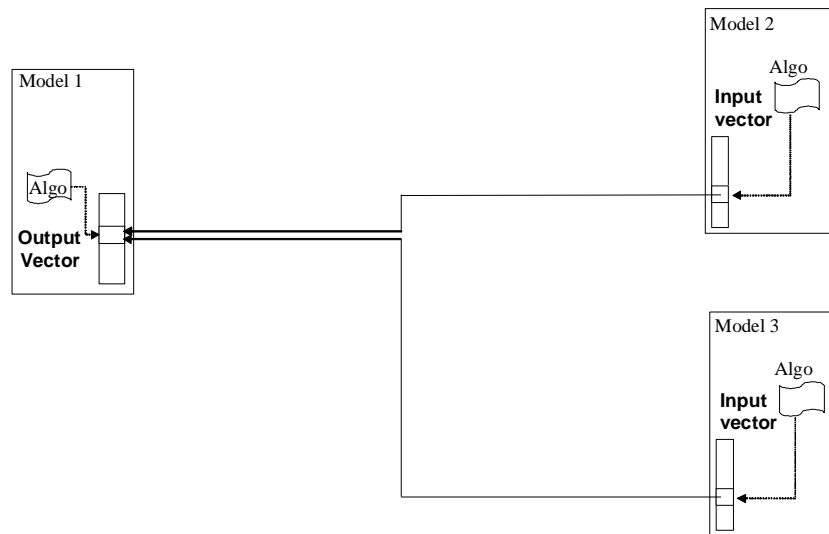


Fig. 3 :Input/Output connection.

### Activation Call-Point to Routine connections

The first model activates its call-point with optional parameters. All connected routines are executed, and the caller may receive an array of responses from activation. A watcher may be attached to this type of connection, to spy the traffic routines (see Fig. 4 hereafter).

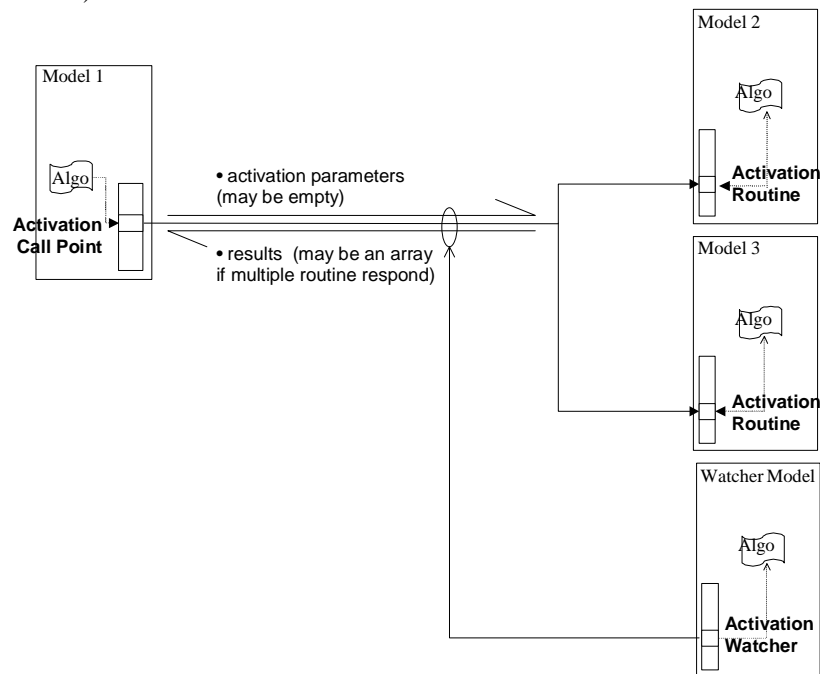


Fig. 4 : Activation Call-Point to Routine connection

### Bus Call-Point to Routine connections

The principle is the same as simple activation, but a 64 bits integer parameter is used to select the destination routine(s) called. Each destination routine subscribe to a (mask, value) couple (see Fig. 5 hereafter).

A routine is selected if :  $(\text{address} \& \text{mask}) == \text{value}$ .

Very efficient algorithm is implemented in the platform: in most case, a bit shift then a mask is applied to the address. The result is used to select the list of bus routines and watchers routine in a table.

A routine may change its (mask, value) subscription. In this case, the tables of this bus connections are reinitialized at next bus call.

NB : the platform doesn't simulate real bus delay, it should be simulated in the model algorithms.

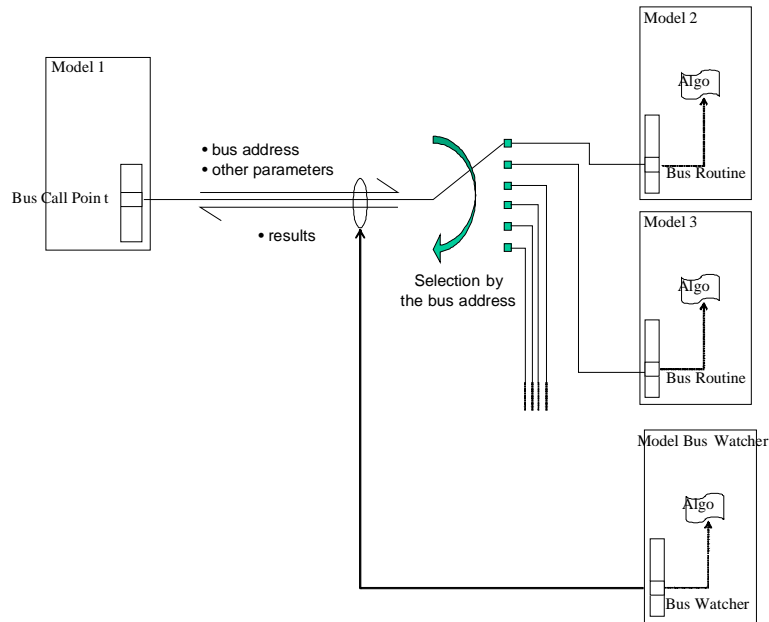


Fig. 5 : Bus Call-Point to Routine connection

## PRINCIPLES OF MODELLING EQUIPMENT HARNESS WITH K2 CONNECTIONS

This paragraph describes the main communication links available and how they are implemented in Thalassim. For each communication type, we provide a real example. Nevertheless, to avoid overloading the diagrams, these samples are not exhaustive : the diagrams contain only the most significant interfaces.

### OBDH bus

The OBDH data bus is a digital time multiplex system composed of two one-way bus lines, the interrogation bus and the response bus. Transmission of information is controlled by the BC and occurs in a fully duplex manner. The interrogation bus provides a dedicated transmission path from the BC (SMU) to all remote terminals. The response bus provides a transmission path from the remote terminal to the BC. The OBDH bus is implemented by using the K2 bus routine / call-point mechanism.

Each bus connection between 2 equipments is implemented by :

- A bus call-point in the master equipment model (sender).
- A bus routine in each slave equipment model (receiver).
- The interrogation word is the K2 bus address parameter.
- The response word is returned within the result of the K2 routine.
- The sender tests the number of routines reached, zero mean no-response, one is ok, more signal a bus conflict.
- Each slave subscribe a couple (mask, value) corresponding to the real parameter address.

Fig. 6 gives an example of an OBDH bus with the Satellite Management Unit (both controllers A and B) and some equipment units (PCU, ...).

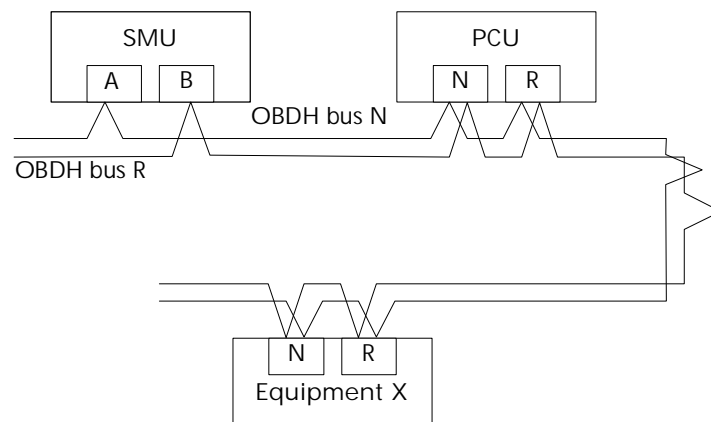


Fig. 6 : OBDH bus topology

The Fig. 7 shows how these equipments are connected together with K2 mechanisms :

- Each equipment becomes a model.
- Links between SMU and equipment units become links between a Bus Call-Point and Bus Routines.

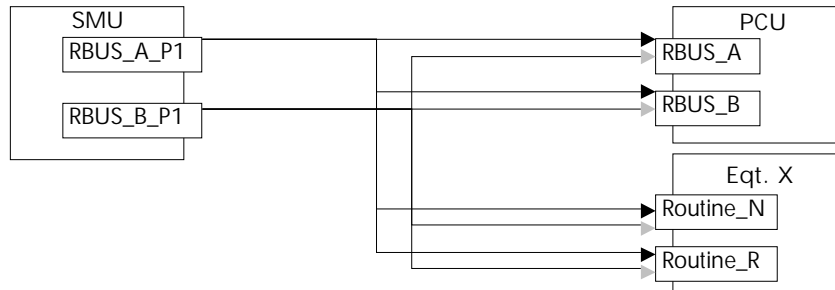


Fig.7: Principles of simulation of the OBDH bus

The SMU model contains :

- 1 bus call-point for each OBDH bus interface (platform OBDH nominal, platform OBDH redundant, PLDIU north OBDH nominal, PLDIU north OBDH redundant, etc...).

The equipment models contain :

- 1 bus routine (activation routine with an address) for each OBDH bus interface (nominal and redundant) connected to the relevant SMU bus call-point. When the equipment contains both nominal and redundant blocks, each block is connected to both SMU nominal and redundant call-points.

### 1553 bus

As for OBDH bus, the 1553 bus is implemented by using the K2 bus routine / call-point mechanism, thus with activation routine (with an address) / call-point (of bus type).

### Clock and synchronization interfaces

As for the previous interfaces, the clock is provided by the SMU, within 2 clock modules (nominal and redundant). Each module provides 3 signals : 1 Hz, 10 Hz and 1 kHz.

Thus the SMU model contains 6 activation call-points. A model needing a clock signal has to implement an activation routine (or 2 if redundancy is required) and connect it to the relevant activation call-point of the SMU.

No specific information is sent with this activation.

### System alarms

The alarms are implemented by using the K2 activation mechanism. Each alarm is an activation call-point / routine.

The information sent on this activation is a boolean data :

- Value True when the alarm is set on.
- Value False when the alarm is set off.

### Low and High Level matrix generator command

The matrix generator command is a system of multiplexing 16 lines and 8 columns. It allows to address 16 x 8 (128) receivers with only 24 (16 + 8) signals. A receiver is connected on 1 line and 1 column and it reacts only when both signals are sent simultaneously within the pulse duration, after a given minimal duration (see the Fig. 8 hereafter).

Fig. 8 : Low and High Level matrix generator command

We have chosen to simulate every line and column pulse. The receiver has an algorithm to detect if it should react or not. A pulse is simulated by an activation Call-Point in the master model, and an activation routine in each equipment unit. The activation has a parameter, the pulse width, and has no result.

### Digital Relay matrix acquisition

As for the Low and High Level matrix generator command, the Digital Relay matrix acquisition is a multiplexing of lines and columns. The aim is to interrogate an equipment by setting a signal on a line and on a column. If the line signal is propagated to the column, it means that the equipment status is on.

## CONCLUSION

All Thalassim concepts (coupling and exploitation of the Satellite Data Base, automatic assembly of simulator), principles of modelling (avionic buses, HW interfaces equipment, ...) and components (K2 plate-form simulator, tools, ...) are validated and qualified.

Thalassim allowed:

- To create at low cost the instances of all Spacebus programs on going.
- To improve the competitiveness for the Avionic Validation.
- To reduce in a significant way :
  - The maintenance cost of the avionic test facilities (savings>50%).
  - The number of tests run on the avionic hardware test bench.
  - The risks of schedule slippage in avionics validation.
- To have an avionic test capacity in accordance with the commercial sales scenario without any additional hardware unit investment.
- To eliminate anomalies due to the real hardware, and corresponding schedule delays.
- To provide an alternate solution in case of failure of an electrical model used in the avionic test facilities and to avoid a single point failure in the avionic validation process.
- To guarantee the in orbit support in case of in-flight failure during the 15 years lifetime of each spacecraft.
- To do a significant technological breakthrough in our :
  - Dynamic Spacecraft Simulator architecture.
  - Electrical Ground Support Equipment architecture.
  - Avionic functional validation process.

The use of Thalassim in the Functional Chains Validation (FCV) process is now generalized to all spacecraft in development. Thalassim is also becoming a simulator product line used at Thales Alenia Space and supports a large range of facilities:

- High Fidelity Simulator (HFS),
- Software Verification Facility (SVF),
- Platform Simulator (SimPF): equivalent to Functional Validation Test bench,
- Avionic Test Bench Simulator (SimATB),
- Spacecraft AIV Simulator,
- Hybrid Simulator (HWIL) : closed loop test with a real unit,
- Dynamics Spacecraft Simulator (DSS): equivalent to Training, Operation and Maintenance Simulator.

Fig.8 shows us the range of Thalassim facilities.

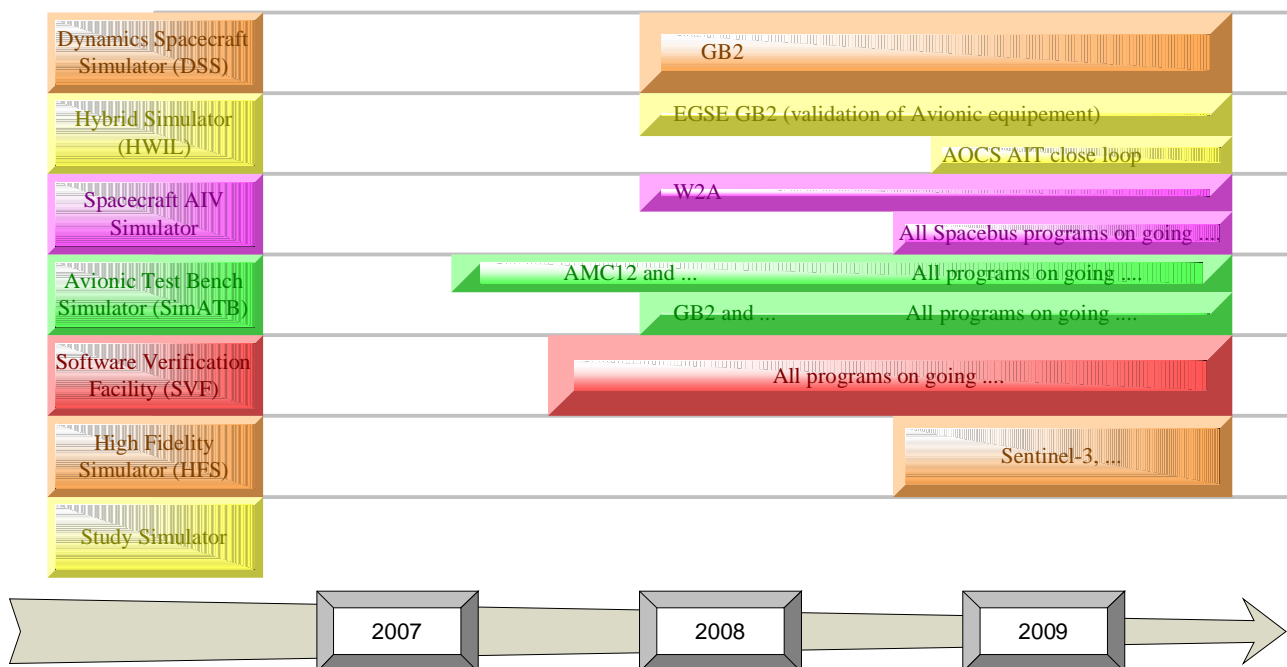


Fig. 8 : Range of Thalassim Facilities