Simulation Of ExoMars2020's Rover Network Using SystemC

Anas Nairi, Julien Plante
Institut Polytechnique des Sciences Avancées
Ivry-sur-Seine, France
{anas.nairi, julien.plante}@ipsa.fr

Nikolai Sinyov, Valentin Olenev, Ilya Korobkov
St. Petersburg State Univervity of Aerospace Instrumentation
Saint Petersburg, Russia
{nikolai.sinyov, valentin.olenev, ilya.korobkov}@guap.ru

***Abstract*—In this paper, we describe the process of implementation, goals and results of our research in the modelisation of ExoMars2020's network using SystemC. We will mainly see that the use of database to store the result of a simulation's running can be efficient in a network involving packets with huge amount of data, and if such kind of network can be made in parallel, then we can increase the performance of its time execution.**

I. Introduction

The ExoMars2020 mission has a goal to search evidence of past or present life on the surface and/or subsurface of Mars. The rover will explore the planets surface and subsurface like its origin of formation and its geological composition, thanks to its list of scientific instrumentation: PanCam, ISEM, CLUPI, WISDOM, Adron, Ma_Miss, MicrOmega, MOMA, and RLS. In fact, with its range of instruments, the rover will map the surface and subsurface in 2D and 3D to localize icy spot, it will list the types of rocks and take very close-up images of them to study the geology of Mars. In addition, it will be possible for it to take samples of the soil.

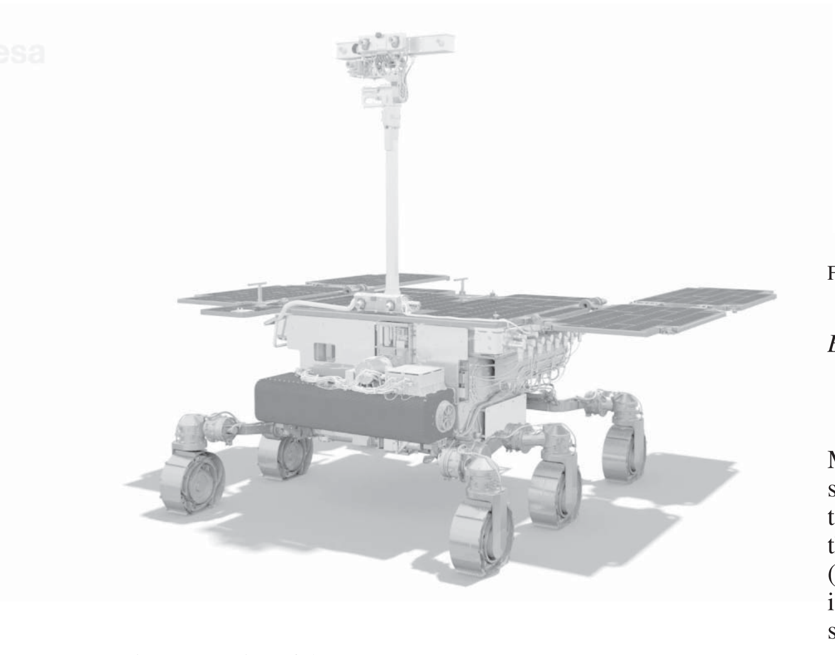


Fig. 1. 3D view of the ExoMars2020's rover

A. *Description of the ExoMars2020's rover network*

The main task of the provided research was to establish a way to implement the network system of the ExoMars2020's rover. To do so, we get inspired with the Fig.2 below, that we can find in [1]. To sum up briefly, we got a network that consists of a SpaceWire router that interconnects all the instruments between each other, making them able to communicate. Moreover, this network got an antenna to simulate the sending of information, resulting from the instruments, from the rover on the Mars' surface to the probe in orbit of the planet. Also, two processors are placed in the rover's architecture, one for general instructions and the other is an image processor, needed because of the amount of images that the rover has to process. And, two memories, one for the mast part of the rover and the other one for the main body part. At the end, data will be sent from equipment, then to the memory, next to the processor which will give instructions to the image processor.

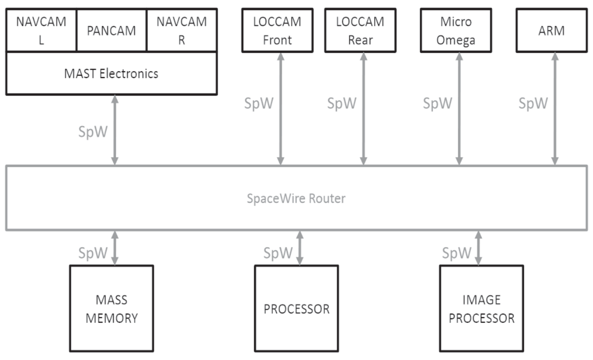


Fig. 2. Data-handling structure of ExoMars' rover

B. *List of ExoMars2020's rover equipment*

1) *PanCam*

The PanCam is used to record 2D & 3D images of the Mars surface, to understand the geological and morphological situation on the planet. In fact, the PanCam is composed of two WAC (Wide Angle Camera), one on the left, the other on the right to have a stereoscopic view of the ground, and a HRC (High Resolution Camera). Also, it operates to produce images in the near infrared and visible wavelength. This device, in synergy with the others, allows the rover to chose the best potential locations site to drill, for example shallow ground that may contains gases or water. The HRC and WAC, both operates with an image resolution of 1024x1024 pixels, the first one in RGB, the second with a multispectral filter wheel.

2) ISEM

The ISEM (or Infrared Spectrometer of ExoMars), as its name means, is an infrared beam that senses the infrared part of the suns light reflection onto Mars surface. With this method used on the rocks near of the rover, it obtains the spectrometer of these rocks and then get information about the geological composition of the planets surface. Indeed, this device helps the rover to chose a potential location to drill in, because the presence of some minerals may indicate good places where to find past life on Mars.

3) CLUPI

The CLUPI, or Close-Up Imager, is a high-resolution camera which goal is to take close picture of rocks to be able to detect each details of these ones. Thus, we can know to what type of rock we are seeing or surrounded with, by analysing the texture, the color, the morphology, etc. Also, it will give information about the original context of possible formation of these rocks, like, geological major events that they experienced throughout their existence. Plus, with this kind of close up pictures, we should be able to detect bio-signatures at the surface of rocks. It is permitted because the CLUPI has an image resolution of 2652x1768 pixels in RGB. It can take picture in the visible wave length, with an exposure time of 1024 seconds.

4) WISDOM

To find if there are some evidence of the past and present life on Mars, we can search in the subsurface, where organics molecules could be shielded from destructive events. In fact, the WISDOM, or Water Ice Subsurface Deposit Observation on Mars, is a ground penetrating radar which will help to notify and describe the type of the shallow surface that is pointed by this one. If an information about a place in subsurface has been processed as a potentially location of organics molecules, the ExoMars drill will then dig in it. WISDOM is working with others instruments to precise if a place in the subsurface is relevant or not by gathering in formation from Adron, PanCam and Ma_Miss. From the Adron, we can obtain information about the composition of a source of water in liquid or ice form. From the PanCam, we get 3D information of the rovers environment, that can help it to better filter locations to dig in. And, because the Ma_Miss is in the drill, it is in direct contact with the subsurface, thus data from this one are compared with those from WISDOM that will permit to set a 3D map of the subsurface.

5) Adron

The Adron is a detector of radiations of neutrons that are sensed from the subsurface of the planet. With the data from WISDOM, it permits to detect the water distribution in the Mars subsurface and the presence of certain elements in this water. Then, the combination of the both instruments data would help to localize the best places to drill in order to find evidences of potential past or present life on the surface of Mars and below it.

6) Ma_Miss

Ma_MISS, for Mars Multispectral Imager for Subsurface Studies, is a spectrometer located in the drill of the rover, used to determine horizontal and vertical composition of the Martian soil.By illuminating the borehole and analysing the reflected light and its spectrum, it will be able to gather information about the distribution of minerals, especially of water-related ones, to search potential indicators of life. It will work alongside the three other spectrometers (RLS,MicrOmega,MOMA), being specialised in studying unexposed material, and in collaboration with WISDOM and ADRON to choose interesting drilling location.

7) MicrOmega

MicrOmega is an infrared spectrometer made to identify composition of Martian soil samples at a grain scale, after their gathering by the drilling system. It is similar as RLS and MOMA in this way, since the three spectrometers will study samples collected by the drill. The infrared study of the samples is adapted to find evidences of past or present carbon and water presence. It uses an infrared hyper-spectral microscopic imager to acquire the spectrum of a 250×256 pixels square ($5 \times 5\text{mm}^2$) for 320 wavelength, between 0.95 and 3.65 m. Thus having a maximum of 20480000 bytes to transmit.

8) RLS

RLS uses the Raman effect to find life signatures in Martian soil samples in a non-destructive way. The measurements carried out by the RLS will be performed as described with in the ExoMars Rover Reference Surface Mission, which includes six experiment cycles (with two samples each,one extracted from a surface target and the other at depth) and two vertical surveys (with five samples each extracted at different depths). It generates information about a 2048×512 pixels of 15 m, totalling a surface of approximately 30.7×7.7 m^2, and 1048576 bytes.

9) MOMA

MOMA (Mars Organics Molecule Analyser) is an instrument designed to detect organic molecules in spots of interest detected by the collaboration of RLS and MicrOmega, thus providing extremely precise analysis of Martian environment, and great information about potential origin,evolution and distribution of life on the planet. To do so, it will study samples gathered by the drill, as well as analysing the gases of the Martian atmosphere. It features to modes of operation: Gas Chromatograph-Mass Spectrometry (MOMA GC-MS) and Laser Desorption-Mass Spectrometry (MOMALD-MS), the first one being used to analyse atmosphere gases, and the second soil samples. No information about size of data produced was found.

C. Related work

Few researches about the modelling of ExoMars2020 rover are shared at the time of writing, especially concerning its network. However, some was found concerning the simulation for other aspects of the rover.

In [4], the authors investigate the use of INRIA SICONOS software to simulate the interaction between the wheels of

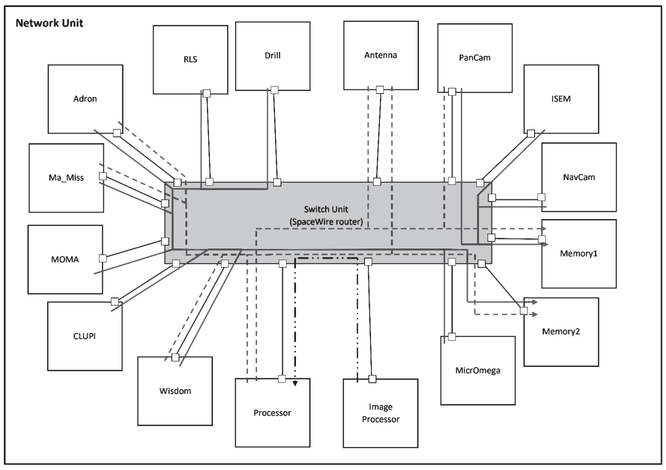


Fig. 3. Single structure of the ExoMars2020's network (caption can be seen with the Fig. 5)

planetary rovers and the ground, thus allowing to study the control law of the rover and its behaviour. They modelled the ExoMars2020 rover using 3DROV, a planetary rover simulation framework, and after adding SICONOS successfully simulated the behaviour of the rover on a planetary terrain.

Another modelling task is presented in [3], focusing on the Panoramic Camera system PanCam, a specific onboard equipment carried by the rover used to acquire images and terrain information after processing. The authors focus on the modelling of the computer vision algorithms behaviour on both single and multiple sites analysis, to determine the distance error induced by the cameras resolution and algorithms themselves.

In the reference [5] and [6], Airbus provides a top view model of the ExoMars' rover to see the placement of the electronic box and the cables that are connecting the instruments. This kind of model can be used to see at the end if the power consumption is okay, and also to test equipment like radars in some real-simulated test.

A lego model had been made (see reference [7]) to choose what is the best way to drive the rover from its lander. To do so, they modelled, with a lego structure, the functionalities of the rover to navigate, they put tiny little cameras. Then, they placed this lego-rover in the CNES Toulouse, and from the ESAs ESTEC in the Netherlands, 1000 km away, they tried to drive in a secure way the rover to get it down from its lander.

II. Main concepts of the project and motivation

We choose to simulate the environment of ExoMars2020 because, as we can see in the Fig.2, it got an architecture where the instruments of the rover are connected thanks to only one SpaceWire router. In fact, it contains equipment that need to work and communicate with each others, memories, an antenna, and processors. That makes the implementation with SystemC useful, cause to instantiate any kind of equipment, we use one module customised for this purpose, which makes flexible the execution of the simulation. Also, we could manage some aspect of the system thanks to the SpaceWire protocol communication, like the speed of transmission of bits and packets, the RMAP protocol to request writing or reading data.

A. *SpaceWire*

SpaceWire is a data-handling communication network developed for spacecraft. It is used a lot in spacecraft nowadays because of many aspects. It is easy to implement in order to obtaining high speed of data's transmission. It is appreciate for its simplicity, its low consumption of energy, and its low cost of implementation. It permits to interconnect equipment between each other, to make them communicate by sending data with a speed from 2 Mbits/s and up to 400 Mbits/s.

Our project features the modelling of the network and packet layer of the SpaceWire specification, as requested, but also a part of the exchange layer. The packet layer covers the use of packets to transmit data from one node to another, with possible routing on the way. Also, we partly modelled an exchange level. We only have the transmission by frames of packets, allowing for transmission delay simulations and transmission error modelling, due to some possible incident radiation. It also made us think about the problem of data collision, since we used bidirectional channels, this forced us to create some security mechanism. But, we did not implement the Error End of Packet (EEP) and error link recovery at this level.

B. *RMAP protocol*

The only transport-layer protocol implemented is RMAP, since we found out that ExoMars2020 uses this protocol.

The RMAP protocol consists of two modes in write and read types. One is the command, that is sent from a node A to a node B, when these two need to communicate together. It mainly handles the sender and receiver logical address, the command that is used like the type used and if acknowledgement are taken into account, the CRC result used to know if the packet has been changed or not, the memory's address in which we want to write or read data to/from, the length of the data, and of course it contains the data sent or read. The other is the reply of the node B which receives the command, it informs the node A if the whole packet has been changed or not, thanks to the value of the status. In that way, it gives the possibility to monitor the error's data bit.

As a matter of fact, this protocol permits to know which node wants to read to and write into another one's local memory. But, note that in our case, ExoMars2020's instruments use the protocol to read to and write into the corresponding part's memory, because we thought that it would correlate more to the structure seen in the Fig.2.

C. *SystemC version 2.3.2*

SystemC is a library of classes in C++. Thanks to this one, we can use a lot of powerful tools in the oriented-object language C++. It contains primary channels that are mainly used like signal, fifo, mutex, or we can simply customised our own channel. Also, it got a lot of structural elements like modules, ports, interfaces. Plus, we can use many data types. Moreover, it got an engine simulation that allows to use

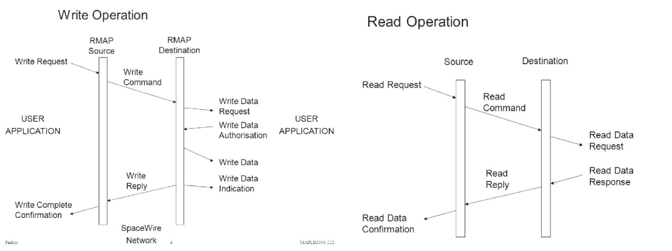


Fig. 4. Write and read operations of the RMAP protocol

processes and events. Actually, it permits to model material and numerical systems in a precised and detailed level of design. One useful tool of SystemC to debug and fix our code is the possibility of creating tracefile, that permits to monitor the state of signals in time during the simulation.

The fact of using C++ language for SystemC is useful in this respect, because in our project we could use other libraries or extensions like OpenMP, SQLite, msbuild, WindowsAPI, QtCreator language and also the C++ standard.

III. RESEARCH RESULTS

A. SystemC : ExoMars's equipment modelisation

As said before, SystemC allows to build and customize our own tools. In our case, we used a custom class named `Node` to simulate the presence of the different onboard equipment of the rover. The use of this class permits its object to use functions to send and write data to another one, that is managed by the RMAP protocol.

Firstly, there are three type of daemons, which are the three main type of threads of this module. We named them daemons because it usually describes threads that run in the background, handling every thing important. The main one is `receiver_daemon`, in charge of handling reception of packets, and choosing what should be done with them. It is written to receive RMAP packets, and sending back replies if needed. This daemon is only launched once, and runs during the whole simulation, which is not necessarily the case of the others. `sending_daemon` and `generating_daemon` are very similar, the first one being in charge of sending RMAP commands, and the second one generating data directly to the node memory. Any number of these two can be spawned, according to the fields "Connections" and "Generations" of the config file. They can also finish before the end of the simulation if their role are finished.

Secondly, the function `send` allows to send a packet of any type, and waiting for a reply. If the reply has a positive status(1), the function ends, and if the status is 0, the packet is sent again, until a positive status is received, allowing for safe communication and error recovery. There is no `receive` function currently, since we only use the reply as acknowledgement packet, but we could create a `recv` function that sends an acknowledgement back, for both commands and replies.

Finally, a function `send_raw` and `recv_raw`, at the lowest level, each one having as parameter one packet and which are respectively in charge of sending actual data into the port and receiving actual data, determining the type of packet and accordingly create the correct packet before returning. One interesting difficulty for the `recv_raw` function was that RMAP uses different header structures for different actions, and we had to determine the type of the packet on the fly, while receiving, to correctly distinguish the end of the header, and the beginning of the data.

This module also features regular logging and database logging, which is not its first role, but still key points, especially for debugging and analysis of the model.

B. SystemC : packet general structure

Another custom class is the description of the different types of packets sent in the two different modes : read and write command, read and write reply. According to their structure, we respected as much the integrity of these packets.

This class aims to simplify communication throughout the network. It contains two vectors, one for the header and one for the data, since each part got variable sizes in the RMAP protocol, as well as two checksums (CRC), for the header and the data. Data can be inserted or extracted as it would be done for a stream, using operators << and >>, which allows to compute the CRC on the fly, as it would be done in a real equipment. But this interface is not always handy, and we needed to add functions about the end of the header, because of the variable header size of the RMAP protocol, and maybe the class would need to provide simpler interface.

That being said, it is very useful to contain packets, simply more than using FIFO, and can be printed in a clean way through the over loading of operator<<.

C. SystemC : SpaceWire router

`SwitchUnit` is designed to mimic a simplified version of the SpaceWire router. Some interesting features are available with this module, such as the table of commutation (also named routing tables, or switch matrix) and wormhole routing. It allows this module to be fully adaptable: the number of I/O ports is determined at runtime, specified in the routing table passed in the constructor of this module. At the beginning of the simulation, a thread `init_thread` is launched, and spawns a `port_processing` thread for each port. These spawned threads are in charge of handling every packet that enter in their ports, and routing it according to the table of commutation of the current `SwitchUnit` instance. To ensure that packets don't mingle, `SwitchUnit` associates to each port a mutex, locked at the beginning of packet transmission through this port, and unlocked at the end. We used a custom mutex that simply inherits from the mutex already implemented in the SystemC library and add the public member function `unlock_event`, allowing to wait for this mutex until its unlocked.

To store the information passing into the router, we use vector from the C++ standard. This class also got a `connect` function, that allows to connect a node to the router through a channel, and automatically modify the routing table to match the new connection. The only limitation here is that we did not write any connect function to link two routers, but we did not need this at the time of writing, since ExoMars2020 contains only one router. But if we need to develop a more complete and versatile simulator, this could become a need.

D. Configuration file

Because of the lack of precise documentation on the network structure of ExoMars2020 rover, the modelling of this network can only be approximate. A configuration file was thus used to describe every SpaceWire node and channel, to include any new information about the network structure easily. Using the JSON format, the interaction between this configuration file and the SystemC model is implemented cleanly, allowing for good flexibility in the description of the network. This configuration file is also used to split the simulated network in independent parts, preparing for the parallel implementation of the simulation.

E. Database log-system

To increase debugging possibilities and to better handle and manage the save of information that has been sent through the simulation, we added a logging system. It was at first under the form of a simple text logging. The information printed to the standard output was also written to files, with one log for each node, resulting in a less mixed output. But this was not very practical, since analysis on these files was tedious, because of no formal format used. This is why we decided to investigate database logging. We chose to use SQLite as database, since its use was much simpler and more adapted than MongoDB, the other database client investigated. Indeed, we wanted to use database as a handy file format to store our information, which is easily allowed by SQLite, that stores everything in one file. MongoDB on its side uses the client/server paradigm, and would have forced to go through the localhost, bringing complications. We successfully logged packet transmission using SQLite, and the ability to perform SQL requests proved its interest, allowing us to find bugs, and permitting custom analysis of what happened during the simulation after its end.

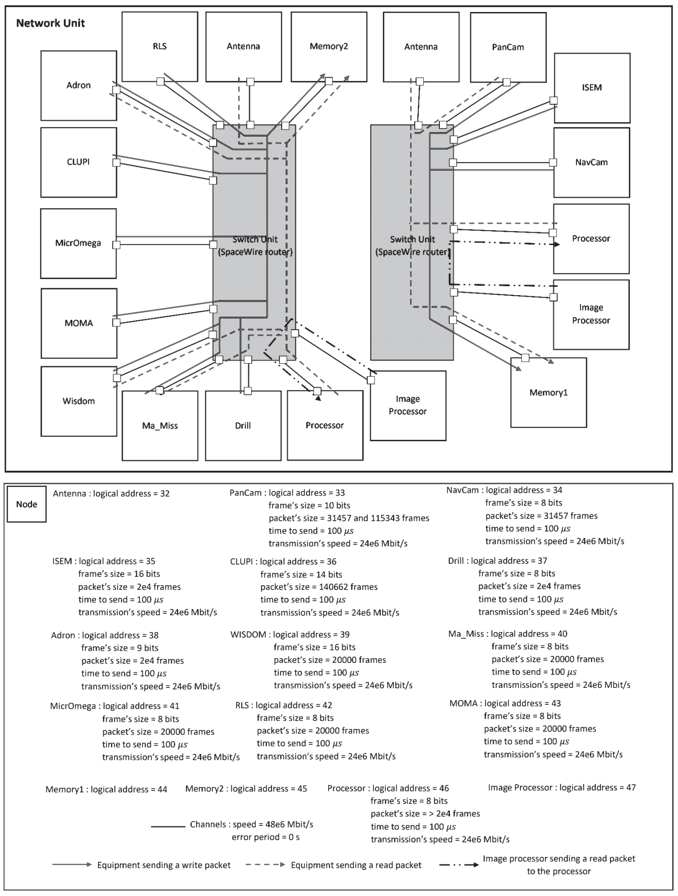
F. Parallel network model

One side-task was to make our single model parallel. In fact, in that way we can see if the rover's network using the SpaceWire communication can be improved using parallel processes. As the rover is already composed of two parts, it is intuitive to make them run in parallel.

Indeed, we transmit in the network big packets, and the simulation began longer and longer as we added features. An idea to improve the time execution of the simulation is to run the two independent parts of the network in parallel, hoping for a performance increase.

To do so, we described the two distinct parts using a configuration file, and used preprocessor directives alongside preprocessor definitions to compile two different executables. Each one of them is then able to simulate one part of the network. For that, we use the Windows API, and the function to run the two parts concurrently.

It can be important to know what results we obtain, if we can make in parallel any type of SpaceWire network. It can give us some interesting results in one case, but in another one it can make the execution time slower. This can help for future SpaceWire implementation in spacecraft.



Antenna : logical address = 32

PanCam : logical address = 33
frame's size = 10 bits
packet's size = 31457 and 115343 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

NavCam : logical address = 34
frame's size = 8 bits
packet's size = 31457 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

ISEM : logical address = 35
frame's size = 16 bits
packet's size = 2e4 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

CLUPI : logical address = 36
frame's size = 14 bits
packet's size = 140662 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

Drill : logical address = 37
frame's size = 8 bits
packet's size = 2e4 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

Adron : logical address = 38
frame's size = 9 bits
packet's size = 2e4 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

WISDOM : logical address = 39
frame's size = 16 bits
packet's size = 20000 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

Ma_Miss : logical address = 40
frame's size = 8 bits
packet's size = 20000 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

MicrOmega : logical address = 41
frame's size = 8 bits
packet's size = 20000 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

RLS : logical address = 42
frame's size = 8 bits
packet's size = 20000 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

MOMA : logical address = 43
frame's size = 8 bits
packet's size = 20000 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

Memory1 : logical address = 44

Memory2 : logical address = 45

Processor : logical address = 46
frame's size = 8 bits
packet's size = > 2e4 frames
time to send = 100 μs
transmission's speed = 24e6 Mbit/s

Image Processor : logical address = 47

Channels : speed = 48e6 Mbit/s
error period = 0 s

Equipment sending a write packet

Equipment sending a read packet

Image processor sending a read packet to the processor

Fig. 5. Parallel structure of the ExoMars2020's network

G. Result of the parallel model

One of the problems we address in this paper is the parallelization of the single model network simulation to increase the modelling speed. Approaches are developed to make in parallel the SystemC kernel itself (reference [2]) but are not shared for now. In the case of our ExoMars2020's singel model, the network can be separated in two independent parts, and the solution chosen to make the simulation parallel, is to create two different simulations, containing each one a part of the network. Since the parts are independent, the results of both simulations are still relevant, and are the same as the results obtained using the sequential implementation of the model. The consistency of the results is ensured by the configuration file, the same being used for each implementation, but interpreted differently to handle the different parts. An increased performance is observed using the parallel model without database logging, as well as why no significant performance increase is noticed when using database logging.

As seen on the I, we can notice on a quad-core machine an increase of performances by a factor 4, only by running two

TABLE I. COMPARISON OF THE TIME EXECUTION'S RESULTS IN SEQUENTIAL IMPLEMENTATION AND PARALLEL IMPLEMENTATION

	Not using SQLite	**using SQLite**
Sequential implementation	24.07 s	54.19 s
Parallel implementation	6.28 s	50.47 s

sub-simulations on two different cores. We expected only a factor 2, since the network is split in only two different parts, and the reason for such performances is still to be investigated.

H. Simulation outputs

Multiple solutions were investigated to gather information on the model during the simulation:

- Console output
- Logfile
- Database
- Tracefiles

Console output and logfile are easily implemented, but lack clear structure, and are thus difficult to analyse afterwards. Waveforms proved to be useful to track errors but are also difficult to analyse automatically. Finally, databases feature deep analysis possibilities using SQL requests, and are adapted to store the results of the simulation. SQLite was at first chosen to handle our database but turned out to be poorly adapted to parallel computation, since parallel access to the database is forbidden, to avoid data inconsistency. The use of other database management systems allowing parallel use of a database is still to be investigated, to feature performance improvement when using the parallel implementation of the model.

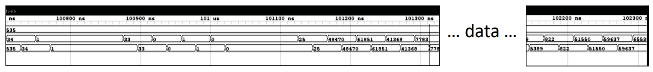


Fig. 6. Example of waveform of a packet received by a node, resulting from a write operation

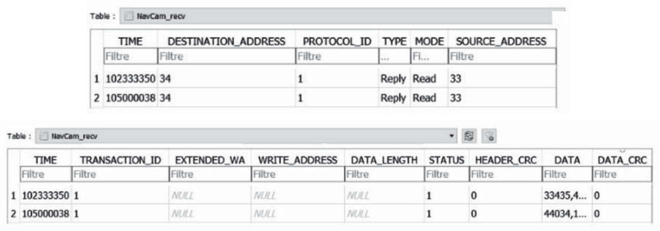


Table : NavCam_recv

	TIME	DESTINATION_ADDRESS	PROTOCOL_ID	TYPE	MODE	SOURCE_ADDRESS
	Filtre	Filtre	Filtre	...	Fi...	Filtre
1	102333350	34	1	Reply	Read	33
2	105000038	34	1	Reply	Read	33

Table : NavCam_recv

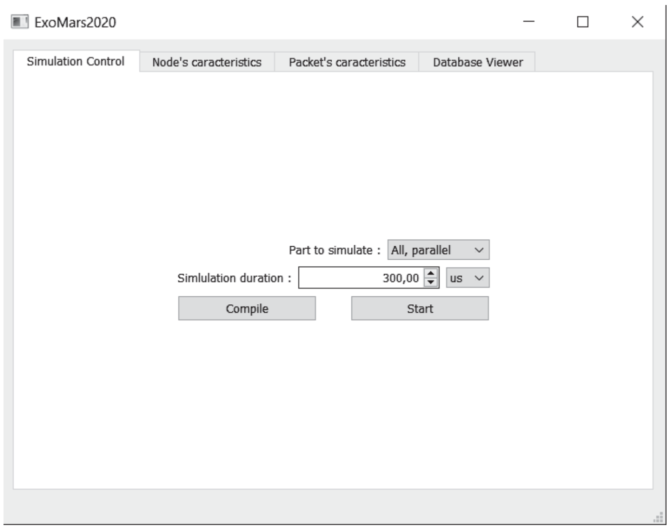
	TIME	TRANSACTION_ID	EXTENDED_WA	WRITE_ADDRESS	DATA_LENGTH	STATUS	HEADER_CRC	DATA	DATA_CRC
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	102333350	1	NULL	NULL	NULL	1	0	33435,4...	0
2	105000038	1	NULL	NULL	NULL	1	0	44034,1...	0

Fig. 7. Example of database display that represents two read replies

I. Graphic User Interface

We chose the Qt framework to develop it, since it is one of the most developed widget toolkit with GTK+, and seems easier to use than the last, since it is written in C++ and thus does not need bindings, which is the case for GTK+. Moreover, the existence of QtDesigner allows us to easily design our UI and quickly obtaining interesting results. At the time of writing, it permits to generate the configuration file, compiling and starting simulation of the different parts of the network, and also analysing the results thanks to a simple SQLite database viewer.

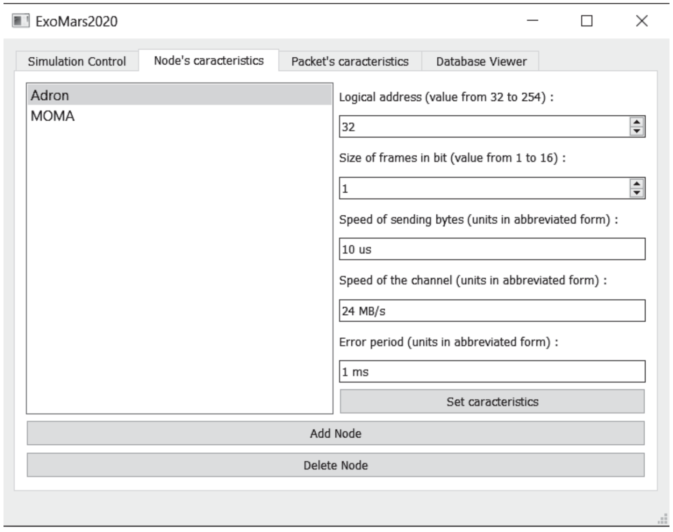
Because of the lack of time, we still have some improvements like reading from existing configuration file to load network parameters, network part management when editing nodes, and possibility to choose from which database we want to read, as well as a refresh button for the database viewer. Further possible but hypothetical improvements could be adding a graphical representation of the network, and a graphical editor, instead of parameters only, and real time animation of the graphical editor during the simulation, in order to representing and also to make in parallel other kind of SpaceWire network communication similar to ExoMars2020's rover.



Part to simulate : All, parallel

Simulation duration : 300,00 us

Fig. 8. First GUI's tab - Simulation control



Logical address (value from 32 to 254) : 32

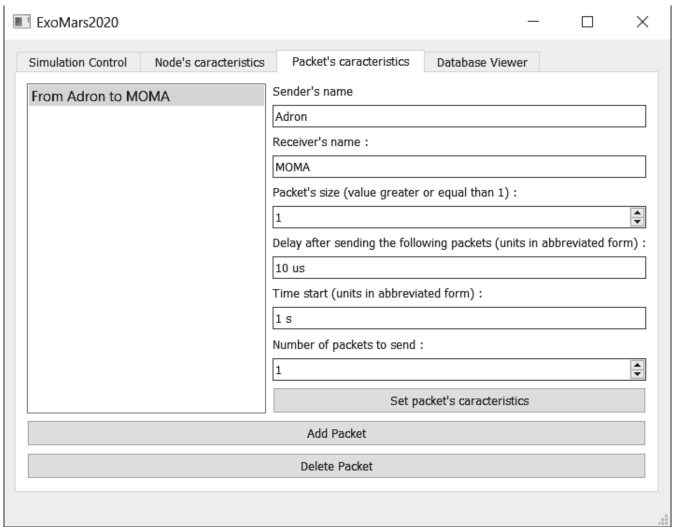
Size of frames in bit (value from 1 to 16) : 1

Speed of sending bytes (units in abbreviated form) : 10 us

Speed of the channel (units in abbreviated form) : 24 MB/s

Error period (units in abbreviated form) : 1 ms

Fig. 9. Second GUI's tab - Specification of a node's characteristics


ExoMars2020

Simulation Control | Node's caracteristics | Packet's caracteristics | Database Viewer

From Adron to MOMA

Sender's name: Adron

Receiver's name : MOMA

Packet's size (value greater or equal than 1) : 1

Delay after sending the following packets (units in abbreviated form) : 10 us

Time start (units in abbreviated form) : 1 s

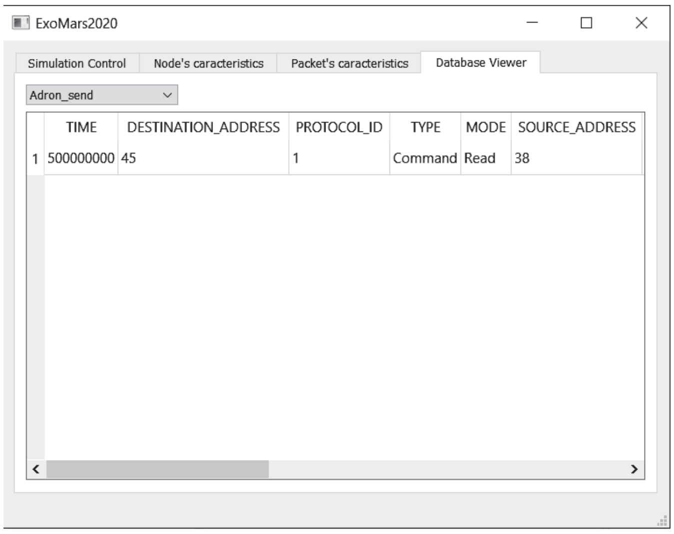
Number of packets to send : 1

Set packet's caracteristics

Add Packet

Delete Packet

Fig. 10. Third GUI's tab - Specification of one or multiple packets' characteristics


ExoMars2020

Simulation Control | Node's caracteristics | Packet's caracteristics | Database Viewer

Adron_send

	TIME	DESTINATION_ADDRESS	PROTOCOL_ID	TYPE	MODE	SOURCE_ADDRESS
1	500000000	45	1	Command	Read	38

Fig. 11. Fourth GUI's tab - Database viewer

IV. CONCLUSION

Building a model, especially in area related to space, is important to test as much general features than specific ones. As we saw, some had been made for the ExoMars2020's rover to describe its internal and external structure, the behaviour of its wheels in contact with the surface of Mars, the process of get down of the lander, and a specific model for one of the equipment of the rover.

Compared to other models, we developed and ran a model of the network of ExoMars2020's rover using SystemC, by implementing the network and packet layer of SpaceWire, as well as a part of the exchange layer, and the RMAP transport-layer protocol. Because little information about the network itself was distributed (most information concerns the scientific research) this model may not be totally relevant, but is a strong basis for future simulations, if more details about the network's structure were to be revealed.

A database log system was implemented as well and proved to be efficient to study results of the simulation, while not being efficient when using the parallel implementation of the model. Other database management systems should be studied to be able to run efficiently in parallel.

Taking advantage of the fact that the network of ExoMars2020 can be split in two independent parts, we succeeded to divide the modelling time by 4, when not using databases. This solution could be used efficiently alongside parallel of the SystemC kernel, to take advantage of all cores.

We are still working on a GUI to configure the network and adapt to further information about the network, and study the results of the simulation.

ACKNOWLEDGMENT

In the situation of doing an international semester in the university SUAI at Saint Petersburg in Russia, we (Anas and Julien) had to complete a R&D project that consists in simulate the network communication of the ExoMars2020's rover. We are two students from the engineering school of aeronautics and aerospace IPSA at Ivry-Sur-Seine in France, in 2nd year of the engineering cycle (end of our study in 2020). It was stimulating to do this project, because we had the possibility to practice and improve our skills and learn new notions in IT programming, plus to develop our culture in spatial engineering. This project had been brought by one of our teacher, Valentin Olenev that introduced and made us the subject of our R&D project. We mainly worked with another of our teacher, Nikolai Sinyov, that explained us some specific concepts that we couldn't understand or when we needed to do major choice to progress in our project. Plus, when needed to do a defense report of our project Nikolai Sinyov and Ilya Korobkov guided and helped us to produce it. Also, our three teachers made some comments and questions during our final presentation in order to fixing some mistakes.

We (Julien and Anas) want to thanks Mr Nikolai Sinyov, Mr Valentin Olenev, and Mr Ilya Korobkov, for their time and their help to realise the project they provided to us during our international semester at Saint Petersburg in the university SUAI, and also to write this publication. Thanks to them, we learned a lot things that will be useful, for both of us, for the rest of our studies and and our business career in companies. We also want to thanks our school in France, IPSA, without who this semester wouldn't be possible.

REFERENCES

[1] Steve Parkes, *SpaceWire Users Guide*. STAR-Dundee, 2012.

[2] B. Chopard, P. Combes, J. Zory, "A Conservative Approach to SystemC Parallelization", *STMicroelectronics - AST, Geneva, Switzerland*, ICCS 2006, Part IV, LNCS 3994, pp. 653660, 2006

[3] Li, D Li, R Yilmaz, Alper, "ESA ExoMars: Pre-launch PanCam Geometric Modeling and Accuracy Assessment.", *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3. 177-182. 10.5194/isprsarchives-XL-3-177-2014.

[4] Vincent Acary, Maurice Brémond, Konstantinos Kapellos, Jan Michalczyk, Roger Pissard-Gibollet, "Mechanical simulation of the Exomars rover using Siconos in 3DROV", *ASTRA 2013 - 12th Symposium on Advanced Space Technologies in Robotics and Automation*, May 2013, Noordwijk, Netherlands. 2013. <hal-00821221>.

[5] exploration esa official website, robotic exploration of Mars, Web: http://exploration.esa.int/jump.cfm?oid=60366.

[6] exploration esa official website, robotic exploration of Mars, Web: http://exploration.esa.int/jump.cfm?oid=60369.

[7] exploration esa official website, space in images, Web: https://www.esa.int/spaceinimages/Images/2016/05/Lego_ExoMars_model.

[8] PanCam team, “The PanCam Instrument for the ExoMars Rover”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[9] ISEM team, “Infrared Spectrometer for ExoMars: A Mast-Mounted Instrument for the Rover”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[10] CLUPI team, “The Close-Up Imager Onboard the ESA ExoMars Rover: Objectives, Description, Operations,and Science Validation Activities”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[11] WISDOM team, “The WISDOM Radar: Unveiling the Subsurface Beneath the ExoMars Rover and Identifying the Best Locations for Drilling”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[12] ADRON team, “The ADRON-RM Instrument Onboard the ExoMars Rover”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[13] Ma_MISS team, “Ma_MISS on ExoMars: Mineralogical Characterization of the Martian Subsurface”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[14] MicrOmega team, “The MicrOmega Investigation Onboard ExoMars”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[15] MOMA team, “he Mars Organic Molecule Analyzer (MOMA) Instrument: Characterization of Organic Material in Martian Sediments”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.

[16] RLS team, “The Raman Laser Spectrometer for the ExoMars Rover Mission to Mars”, Jorge L. Vago. in Astrobiology Vol. 17, No. 6-7.