POLITECNICO DI TORINO

Master’s degree Course in Mechatronic engineering
Master’s degree Thesis

STABILITY AND ACCURACY ANALYSIS OF DIGITAL REAL-TIME SIMULATORS INTERCONNECTION FOR CO-SIMULATION INFRASTRUCTURE DESIGN

by
Sara Hernández Vargas

Advisors:
Luca Barbierato
Andrea Mazza
Edoardo Patti
Enrico Pons

April 2021

This thesis is dedicated to
my family in Costa Rica:
Papi, Mami, Daniel and Nano;
and to my family in Italy:
Fran and Esteban

Table of contents

List of Tables

List of Figures

Abstract

Co-simulation has become one of the most powerful tools to test and validate large and complex power system scenarios avoiding the risk of harming expensive equipment and affecting power system reliability. In fact, co-simulation becomes specifically important to test and simulate innovative functionalities with complex requirements in nowadays electric grids and to evolve to Smart Grid concept.

Hardware in the Loop simulation technique is mostly used for development and embedded system testing. HIL needs real-time simulation models to allow the artificial injection of inputs to the Device Under Test (DUT) and the monitoring of the DUT outputs. But when testing critical devices, the co-simulation infrastructure, as well as the stability of the system, is fundamental to be assured to avoid problems with the connected devices, when outbound signals (due to an unstable system) are sent to the DUT that might cause overvoltage or overcurrent.

The aim of this thesis work is to analyze the main co-simulation techniques and standards to evaluate the feasibility of data transmission between different Digital Real-Time Simulators [DRTS] exploiting high-bandwidth and low-latency protocols (e.g. Xilinx Aurora 8B/10B) to allow Hardware In-the-Loop (HIL) testing in co-simulation infrastructure. The communication latency is a very important parameter to be determined before interconnecting different DRTS, to run complex power system simulations, because it is considered one of the main numerical stability issues. In fact, latency could influence on time-domain accuracy and frequency-domain stability of the solution in fast time-stepped simulation.

In this work, the time delay of communication protocols that enable DRTS interconnection is measured to ensure that latencies will not affect power system co-simulation results. As part of the analysis, this calculation has been compared with other variables to determine if their values affect time delay behavior of DRTS interconnection, such as the amount of data transmitted, and the imposed time step.

Retards are present on PHIL systems due to the delay time from when the electrical signal is transmitted from the real time simulator (RTS) and passes through the power amplifiers (coupled in the real time simulator) until the hardware under test (HUT) actually receives the signal. The retard phenomenon happens also in co-simulation, but the retard is due to delays in the communication from one point to another.

Based on the previous statements, the purpose of the thesis second step is to create a simple electrical circuit to be implemented and split it in two parts running on two DRTS. In this stage, a theoretical analysis was performed to stablish the parameters that ensured stability of the system. To state those parameters, the theoretical analysis was approached by an Interface Algorithm method, which through a set of non-differential equations analyzed the stability of delayed systems. Once established the parameters that assured the system stability, the simulation was performed using a single DRTS (RTDS Technology Novacor2 chassis) exploiting an echo link to experimentally test the behavior of the system. The performed tests on the DRTS showed the limitations of the real-time simulator when comparing it with respect to the theoretical simulations. As conclusion, it is stated the appropriate parameters regarding the time step and circuit specifications that must be followed to set-up a composed co-simulation infrastructure in DRTS.

1 Introduction

With the increasing innovations in power electric system, the necessity of validation of the Power Generation and Distribution Systems is required. Most of the times, power systems are composed by complex systems as the Electromagnetic Transient (EMT) and AC grids that relies on the information and communication technology to control and monitor the system [1].

In the last decade, the power industry interest is targeting on Hardware and IO testing, to consecutively analyse the interfaced plant model reaction which is simulated by prototypal models. The attention is particularly emphasized in power electronic converters and fast acting control and protecting systems [2] where the system model needs to be tested at the same time rate as the real-world physical system. In such cases, the Power Hardware in the loop (PHIL) becomes a necessity. PHIL allows real time simulators to be interfaced with prototypal setups of the devices under test (DUT) or IO.

However, the implementation of a PHIL based model requires a previous analysis of the model stability, due to the communication latencies imposed by the data transmission from the device under test (DUT) to the rest of the system (ROS). The study of the system stability and the frequency domain through Interface Algorithms models have been proposed in the past few year [3], as the damping impedance method (DIM), ideal transformer model (IT), etc. The theoretical analysis aims to identify the stability limits, to narrow the critical study cases to be tested on the simulator.

Some simulation models require to interface sub-systems needed to be simulated at different time step rates. For instance, in [4], the AC grids need a time step up to 500 microseconds, instead the EMT can be simulated in a range between 2 and 50 microseconds. The problem arises when these systems, with different simulation rates or even different time domain modelling (as frequency and continuous time), are needed to be interfaced.

Another difficulty occurs when it is intended to simulate large systems. In [5], a large complex EMT system was needed to be simulated. In this case, the simulation timestep was not multi-rated and the simulation domains were the same, but in such large systems, many times it is required to divide the system in *i)* a subsystem subjected to less changes and *ii)* another one that provides model changing flexibility, due to the system test complexity.

Depending on the type of simulation intended to be performed (networking simulations, EMT system simulation), the simulations can be performed using three main simulator types: time-stepped simulators, event-based simulators and real-time simulators. Normally, because of the characteristics of the networking traffic, the most common simulator to use in networking system modelling is the event-based simulator [6]. On the other hand, since the stepped time simulator work with a fixed time step (in the microseconds order), these simulators are well suited to perform EMT simulations [7].

The main difference between the time stepped simulator and the event-based simulators is the updating system state variables frequency. In the case of the event-based simulator, the system state variables can be updated only when an event is detected, instead in the case of the time-stepped simulators, the state variables can be updated on each fixed time interval. The real time simulator approach is quite different to the previous two, since this simulator type allows to perform tests as if the real process was connected.

The main advantage that real time simulators offer is a simulation as if the device under test (DUT) was connected in real world physical time to the simulator, providing an interface to monitor the rest of the system (ROS) behaviour and the DUT.

The main problems of a detailed solution of many electrical systems (e.g. EMT system and AC systems) is the significant use of computational resources [5]. Therefore, one of the main solutions implemented in the last decade is the co-simulation of two real time simulators allowing to test large systems and maintaining, at the same time, the simulation accuracy.

Nevertheless, in co-simulation, it persists the need of finding a method of synchronizing the simulation tools being used, where again, the communication latency between the involved parts is a fundamental key to assure the system stability thus its reliability. In order to determine the best suited synchronization technique, it is important to determine the time range on which the synchronization is needed. Some protocols like Precision time Protocol can synchronize the simulation network up to 100 nanoseconds but requires a PTP grandmaster (GPS or CDMA). Instead, other protocols, like SNTP, are less accurate (in the range of milliseconds) but do not require a GPS or an atomic radio.

In recent studies, RTDS and OPAL-RT are two real time simulators widely used to implement complex architectures when simulating large EMT systems [5] or even geographically distributed EMT systems [8]. In [5], it was implemented a co-simulated system by the interconnection of RTDS and OPAL-RT, instead in [8] the geographically distributed EMT system was implemented by using field programable gate array (FPGA) with the RTDS. In particular, RTDS has a flexible framework to perform system reconfiguration when modelling or performing control features.

The main purpose of this thesis is to analyze the main co-simulation techniques and standards to determine the feasibility of a real time simulator (specifically RTDS) when interconnecting two different architectures through an echo full duplex link, using a low latency and high band protocol: Xilinx Aurora 8B/10B. The communication latency is a transcendental parameter since it can influence the time-domain accuracy and frequency-domain stability of the solution in fast time-stepped simulations. Therefore, communication latency must be calculated to assure that it will not perturb the power co-simulation results.

The following subsection states the thesis motivation, the goals of this work, as well as the thesis document organization.

1.1 Thesis Motivation

In many research papers, co-simulation of hybrid systems, using different types of simulators, are approached as the co-simulation of an event driven simulator interfaced with a real time simulator. This is the case of [9], where the OPNET (an event driven simulator) was used to simulate the communication network, while the power system simulations were conducted in RTDS, a real time simulator. In [4], the co-simulation of two real time simulators DSP and PC is performed to achieve multi-rate simulation, using also FPGA as the simulation clock controller. But there is not relevant work of co-simulation of OPAL-RT and RTDS that determines the co-simulation capability that these two real time simulators can provide.

Based on the previous observations, the future intention in the Energy Center of Politecnico di Torino is to perform co-simulation of NovaCOR RTDS from RTDS Technologies and OPAL-RT

OP5700. The co-simulation of these two RTSs will allow to test large and complex systems, as well as to evaluate system configurations when connecting the model to a device under test (DUT) in PHIL. The RunTimes of both DRTS are installed in a host workstation that allows set-up and monitoring of the RTSs.

Looking forward to accomplishing the stated future intention of the Energy Center, the main goal of this thesis is the determination of the latency when transmitting data from one port to another in a RTS. The RTS used was NovaCOR RTDS from RTDS Technologies, performing the communication through Aurora protocol. The connection was implemented by an echo full duplex link by connecting ports 23 and 24 of RTDS chassis 2, allowing bi-directional communication through optical fiber cable as the physical layer. The second statement that this thesis presents is the evaluation of RDTS capabilities and its limitations when simulating a communication delayed architecture based on two-parts split electrical system model.

The structure of this thesis is organized as follows. The Technological background reviews the type of power system simulators, the co-simulation techniques, and the proprietary protocols to interface DRTSs. The State of art solution introduces relevant works related with this thesis. The Methodology chapter describes the methods used to develop the test cases. In the Analysis and accuracy stability chapter the RTDS data transmission latency time (through Aurora protocol using an echo link) is calculated and the simple study case is described. The simple study case is analyzed theoretically, regarding the system stability analysis, to continue afterwards with the experimental testing on the RTDS. Finally, the conclusions and the summary of the total work are presented in the Conclusions chapter.

2 Technological Background

2.1 Power Systems Simulators

In Power Systems simulations there might be several models that need to be simulated. These models include from the networking system to the electrical grid or control system. When performing domain specific model simulations, three main categories can be individuated. The classification is based on the frequency of the iterations as the event-based simulators or time stepped simulator. The digital real time simulators possess a different approach with respect to the other two simulators. The main characteristics of the mentioned simulator types are described in the following sub-sections.

2.1.1 Event-based Simulators

The main characteristic of the event-based simulators is that the iterations are seen as simulation events. Normally, a power system simulation is mostly implemented with a fixed timestep while the information network is event-driven, however, power systems can also be simulated as event-driven simulation where the performed iterations are seen as events. Network data events are always randomly distributed, since commonly are simulated as discrete events [6].

Focusing on the common use of the event-based simulators, several studies have been held analysing the feasibility of the simulators to accomplish network or electric grid simulations. OMNET++ is a discrete event simulator intended to simulate electrical networks programmed in C++ in a modular infrastructure. The feasibility of the OMNET++ to simulate also electrical grids was proved in [10], where it was concluded that the simulator can successfully perform simulations in these systems, despite it was not supported initially for this purpose. However, some disadvantages arise in the development phase since it is stated the concurrency of crashes while simulating.

NS-2 is a discrete event Network simulator that is aimed for the simulation of TCP/UDP, routing and multicast protocols. In the application level is stated to cover HTTP protocol, telnet and FTP sources, etc. [11]. NS-2 is programmed in C++ in a modular fashion as well as OMNET++ [12].

OPNET Modeler is a product of PNET Technologies which is also a network simulation software that aims to solve network management issues [13]. This simulator is able to create traffic of telecommunication networks and protocol modelling [14].

Some authors developed simulators since stated that the documentation of the previously mentioned simulators to perform new simulations is incomplete and the learning process is time consuming and difficult. TARVOS [12] is an open source discrete simulator coded in C, intended for research purposes that can be customized and controlled from a high level which facilitates user’s simulations setup.

2.1.2 Time Stepped Simulators

The simulator type is based on the frequency on which the algorithm solver is called. In the case of the time-stepped simulator, the time step is a fixed time interval equal to the step size and is the traditional simulation implementation.

In [15], the time stepped simulation efficiency is discussed, since during low density of events there is a waste of processing time. Therefore, [15] proposes a technique to combine the event-driven method with the time-stepped one to optimize the computing resources. Although, it is important to state that time-stepped simulations are very efficient when event density is high.

Generally, the time stepped-simulators solve the differential equations in a time stepped manner, on which the time step is in the microseconds order [7]. In each step, the algorithm simulates all events that appear in that time interval [15]. This makes them very well suited for simulating electromagnetic transients and in general electrical models. Some simulators that behave the previous described way are: EMTC and PSLF [7].

2.1.3 Digital Real-time Simulators

The digital real-time simulators (DRTSs) serve as a study tool to determine causes and solutions on complex electrical systems involving millions of dollars. DRTSs are very important in industries like Energy, where performing a plant test (for example smart grid application or control paradigms) is not only dangerous but also can damage expensive equipment.

Real time simulation allows to perform setup tests as if a real process would be directly connected to the simulator via a communication system. The simulator is designed to react fast enough to the signals it is receiving [16]. In this way, it is possible to make decisions about the changes in the model and evaluate its consecutive real plant implementation.

The real time simulators have also the advantage that real Hardware can be interface with it and monitor the device behaviour. When real Hardware is implicated, the concept behind it is Power Hardware in the Loop. The main characteristics of a real time simulator are: a dedicated Runtime environment orchestrated by the solver and the guarantee of a specific computation time that is stable [2].

IREQ simulator is analysed in [17], which has a hybrid digital and analogue data acquisition system and an online data processing system. The models were run in a real-time closed loop feedback system. The inputs are read from voltages and currents applied to the simulator dedicated hardware. This study also states that since analogue technologies still require fast transients in real time, the best solutions are based on hybrid models where the analogue components or the passive components are connected to the digital models [17].

In [18], it is evaluated the better choice of a time step in a digital real time simulator developed for relay testing with a single high-performance workstation processor that contains a graphical user interface in the real-time power system simulator.

When designing a simulator, one of the main aspects to take into account is the timestep, because this will define the type of applications that simulator will be useful for. [18] analyses the many factor that affect the timestep. One of those aspects is the frequency bandwidth of the output signal. The bandwidth value is determined by the input wave analysis.

The time step simulation accuracy will also be affected by the network complexity which is directly influenced by the number of nodes in the network. Finally, the simulator architecture and the I/O subsystem requirements are aspects that might increase or decrease the timestep of the simulator.

2.1.3.1 Opal-RT

OP5700 is a real time simulator built by Opal-RT Technologies which simulation software is the RT-LAB. RT-LAB enables rapid prototype since it is an integrated real time software where control system test and HIL simulation can be performed. Opal-RT uses MATLAB (specifically Simulink) as a model editing tool, working as user designing front-end application [19].

The OP5700 architecture is divided in two main sections. In the first section the analogue and digital I/O modules are assembled, and the second section is composed by a multi-core FPGA able to run the models of the real time simulation software. The OP5700 can be connected through TCP/IP Ethernet to a Windows hosting computer to edit and monitor the simulations [20]. Figure 1 illustrates the simulator architecture.

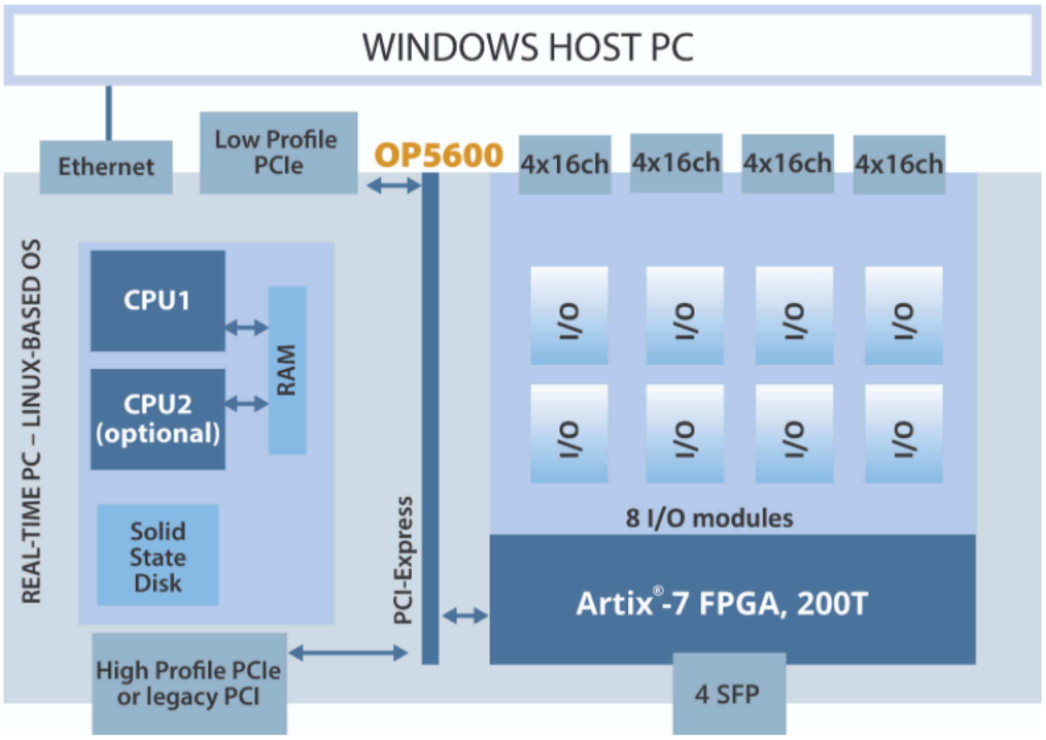


Figure 1. Opal-RT simulator architecture [20].

OP5700 has 8 slots of I/O boards per system. Each analog board has 16 channels and support up to 128 in the system. Each digital board has 32 channels per slot and support up to 256 per system. Regarding the connectivity, supports Ethernet and RS232. The supported communication protocols for Energy systems are S7, Profibus, Modbus, S7, Aurora, IEC 60870-104, DNP3 outstation (slave) and master, ABB PS935 [20].

2.1.3.2 NovaCOR RTDS

NovaCor is the new generation of simulation hardware of RTDS (stands for Real Time Digital Simulator). The RTDS is a simulator used mainly for real time power system simulations [21]. The RTDS cubicles are composed by the racks that host the chassis with the processor cards and the IO cards fed by the correspondence power connections. Each chassis possesses a powerful muti-core processor designed to perform EMT simulations. The chassis also contains communication ports and analogue output channels [22].

Figure 2 shows the composition of each chassis, containing an IBM Power8 processor with 10 cores operating at 3.5 and a first FPGA that handles real time functionality [22]. The RTDS also contains Workstation Interface Card (the other FPGA card) which main task is to perform connection between the RTDS rack and the workstation (that hosts the software able to perform sophisticated graphs through ethernet based connection) [23].

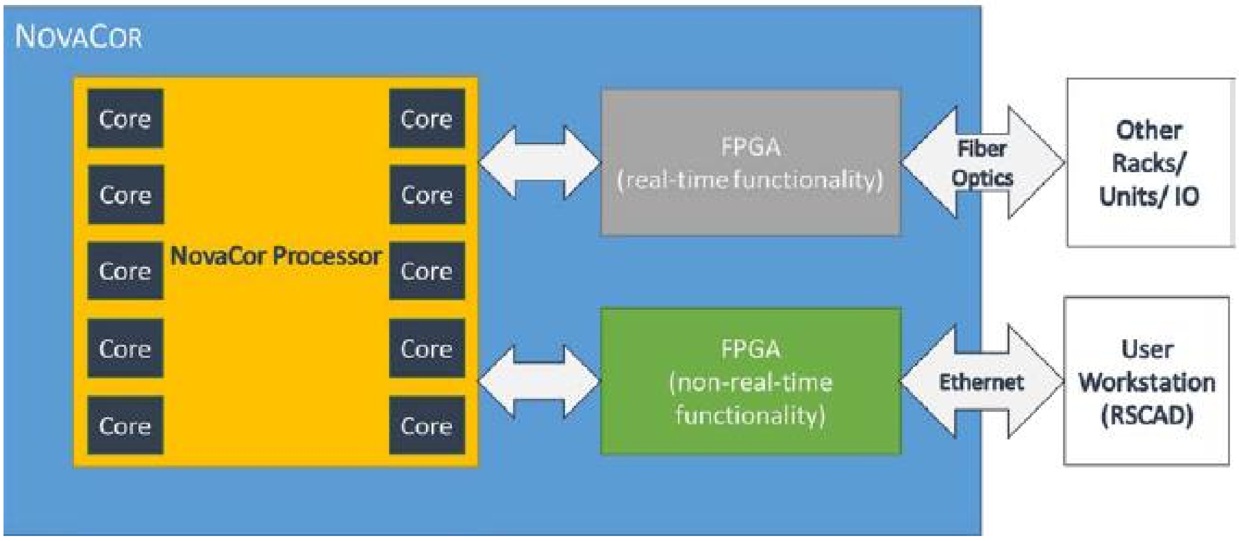
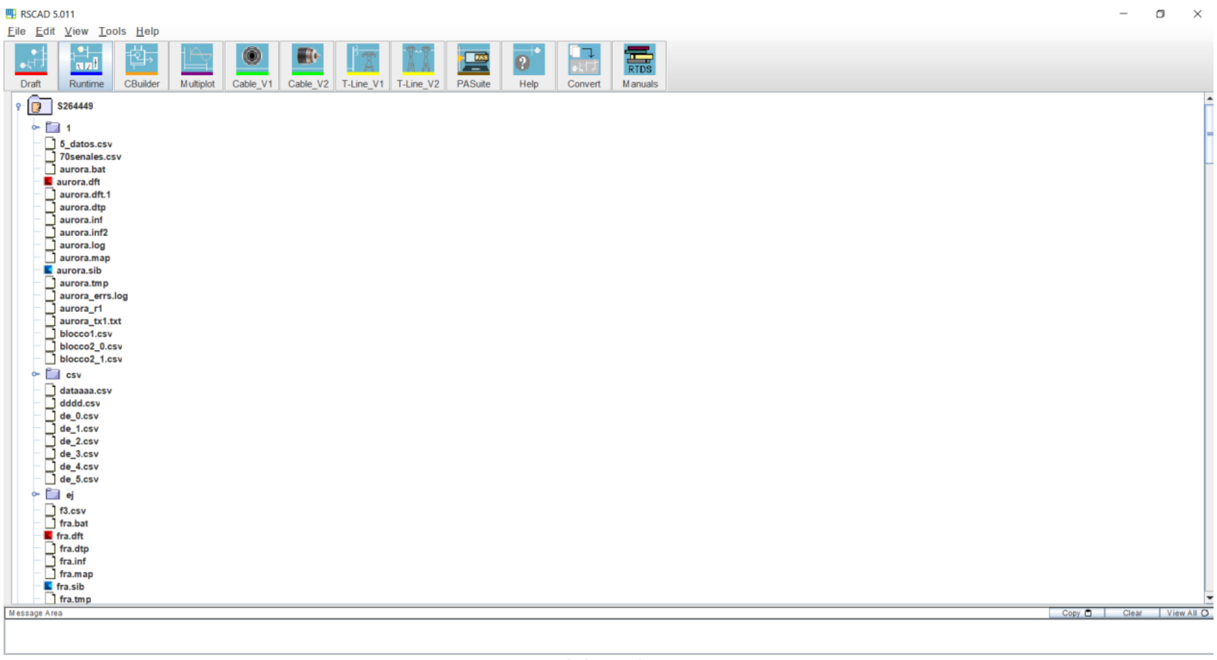


Figure 2. Novacor system architecture diagram [22].

The maximum quantity of IO cards supported per chassis is 160. Each Novacor fiber port supports up to 8 IO cards and each Novacor has up to 24 GT Fiber communication ports, where only 20 are available for IO card connection, and ports from 21-24 are reserved for Aurora communication [22].

RSCAD is the software used for modelling the systems that is also design by RTDS technologies. Through this software, it is possible to develop, compile and run the models. The RSCAD/File manager application is the interface between the developing and running application of the simulator.



```
S264449
  1
    5_datos.csv
    70senales.csv
    aurora.bat
    aurora.dft
    aurora.dft.1
    aurora.dtp
    aurora.inf
    aurora.inf2
    aurora.log
    aurora.map
    aurora.sib
    aurora.tmp
    aurora_errs.log
    aurora_r1
    aurora_tx1.txt
    blocco1.csv
    blocco2_0.csv
    blocco2_1.csv
  csv
    dataaaa.csv
    dddd.csv
    de_0.csv
    de_1.csv
    de_2.csv
    de_3.csv
    de_4.csv
    de_5.csv
  ej
    f3.csv
    fra.bat
    fra.dft
    fra.dtp
    fra.inf
    fra.map
    fra.sib
    fra.tmp
```

Figure 3. RSCAD file manager.

Through the file manager, it is possible to navigate in the existing files, open and modify them. It is also possible to create new documents. The main application used to develop the system intended to be simulated, is the Draft.

The Draft application possess a library that contains, in different classifications, the algorithms to develop the models. The library has two usage modes: the master and the user. The master is the standard library and remains unchangeable, instead the user's library allows to personalize the library. The components are classified in three parts: Power System Component Library, Control System Component Library and the Small Time Step Component Library.

RSCAD/MultiPlot is an application used to process and analyse captured results stored during the simulation. The RTDS Component Builder is the application that allows the user to create new components in both Power and Control systems. It provides an interface for drawing and creating new components.

The RSCAD/RunTime is the application used to simulate the generated draft, once it is properly compilated. In the case options it is possible to set the simulation time and the update frequency.

The supported communication protocols by RTDS are MODBUS server (slave) over TCP, DNP 3.0 slave, IEC 60870-5-104 slave. Regarding the open protocols, IEC 61850 9-2LE or IEC 61869-9 and Aurora are supported [24].

2.2 Hardware-In-the-Loop

The Hardware in the Loop technique requires developing a simulation close enough to reality that models the plant and connects the device under test (DUT) to determine its behaviour by the interaction with the simulated plant. In this case, the system will read real inputs (like sensor signals) and generate simulated output signals that in general are commands to the actuators.

HIL can be used also to determine that the developed software, intended in the future to be run in the plant, won't damage the devices and will work properly, which makes HIL a powerful tool to discover possible bugs or problems both in the hardware and the software. This feature allows the possibility to optimise the product before its production or field implementation.

However, to assure the functionality of the tests performed in HIL, it is critical to demonstrate that the simulated environment is a sufficiently reliable representation of reality.

The verification and validation are the steps that demonstrate the correctness and fidelity of the simulation. Verification is the process on which HIL simulation results are compared with analytical calculation results or results from independently developed simulations. Validation consists on demonstrating that the HIL simulation models are suitable in the operational environment with an acceptable accuracy [25]

2.2.1 Stability and Accuracy Analysis

When performing HIL simulations, [26] stated that all systems can be modelled by dividing them in two parts: the device under test (DUT) and the rest of the system (ROS) represented in Figure 4. Normally the ROS is simulated in the real time simulator and the DUT is implemented by real hardware connected to the ROS through the using of an Interface algorithm (IA). When decoupling the system, delays, errors and non-idealities appear that could derive in the instability or inaccuracy of the system.

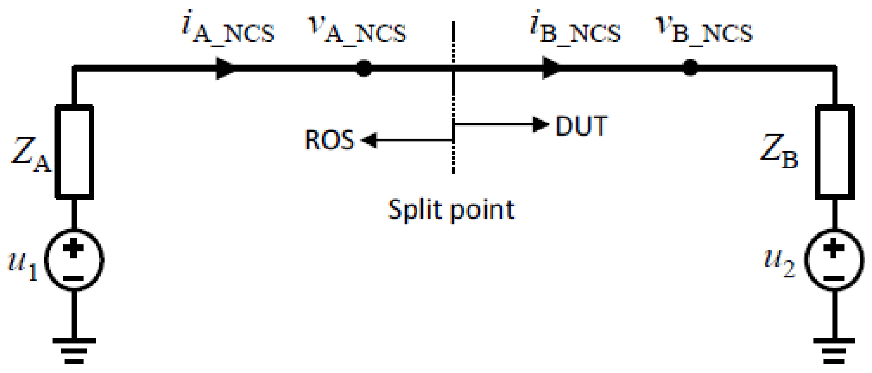


Figure 4. Representation of system split [26].

To develop and test models through the HIL simulation, the stability of the models must be assured. The stability of the HIL system can be analysed using the system transfer functions. To assure the stability, the open loop transfer function must comply with the Nyquist criterion.

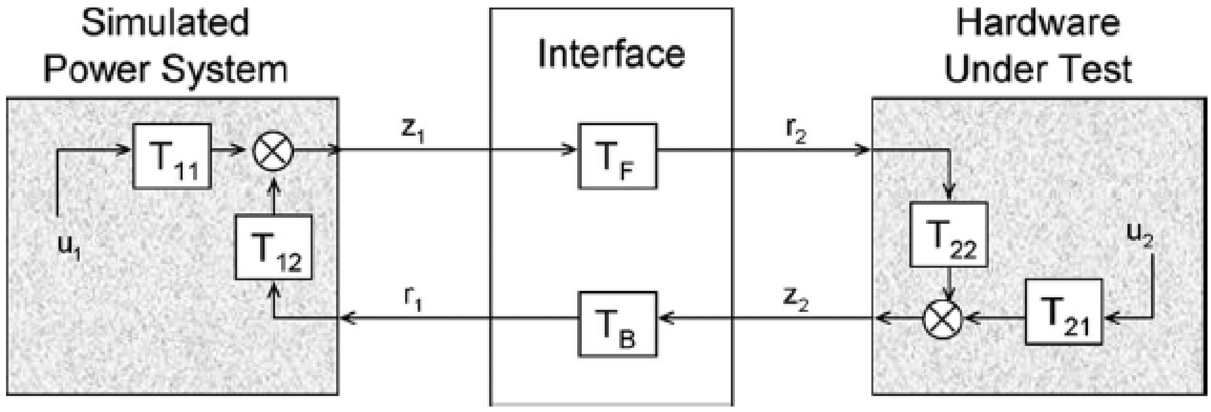


Figure 5. Block diagram of a HIL system [3].

The transfer function from Figure 5 is shown in Equation (5):

$$G_{OL\_ITM} = T_{12}T_BT_{22}T_F \tag{1}$$

2.2.2 Interface Algorithm (IA)

The interface algorithm determines which is the type of signal to be transmitted and how is this signal being processed. In the case of the transmitted signals, the voltage, current or power are very often used variables. Regarding the way the signals are processed, models commonly use the received signals as inputs passing through logic composed of gains, low pass filters, lead lags, etc [3].

2.2.2.1 ITM

The ideal transformer model (ITM) algorithm is considered as the simplest interface model. In [26], the error caused by decoupling the system is lumped in a single time delay Δt. So, the system is modelled as the open loop transfer function multiplied showed in Figure 6.

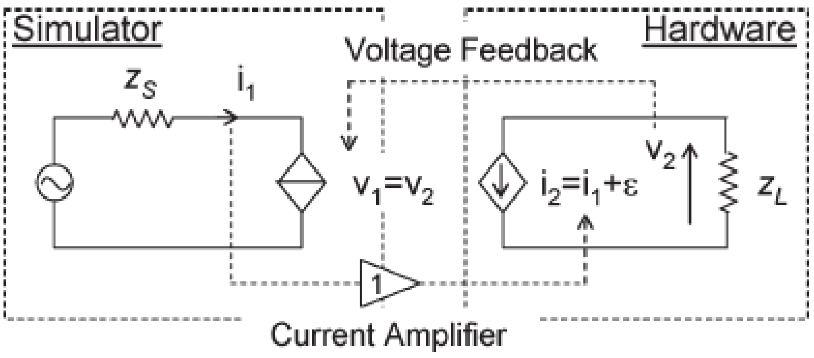


Figure 6. Delay due to decoupling the system [3]

From Equation (1):

- $T_{12}=-Z_S$

- $T_B=1$
- $T_F=e^{-s\Delta t}$
- $T_{22}=1/Z_L$

The transfer function equations that model using the ITM IA for the voltage type:

$$G_{OL_{ITM}} = \frac{Z_S}{Z_L} e^{-s\Delta t} \tag{2}$$

2.2.2.2 Shifted Impedance

In [26], the shifted impedance method is stated by adding a phase shift in the voltage and current of the ROS. Supposing the system is of the form of Figure 6, and that all delays are lumped in a single time delay Δt , the open loop transfer function is described by Equation (3).

$$G_{OL\_SI} = \frac{z_S - z_{SFT}}{z_L - z_{SFT}} e^{-s\Delta t} \tag{3}$$

The drawback of this system appears when implementing co-simulation with HIL, since adding a shifting impedance (z_{SFT}) in a large power system, might result very expensive [26].

2.2.2.3 Damping Impedance (DIM)

The Damping Impedance algorithm exhibits the greatest stability of the broadly studied interface algorithms because of its potential of being adaptively controlled using PHIL in real time simulation [27].

The DIM IA is obtained by adding a damping impedance (z*) when the combination of the ITM algorithm and the PCD (partial circuit duplication) is performed. PCD is another interface algorithm that consist on the relaxation technique intended for large software simulations [3]. Figure 7 shows the implementation of the co-simulation by using DIM as an interface algorithm.

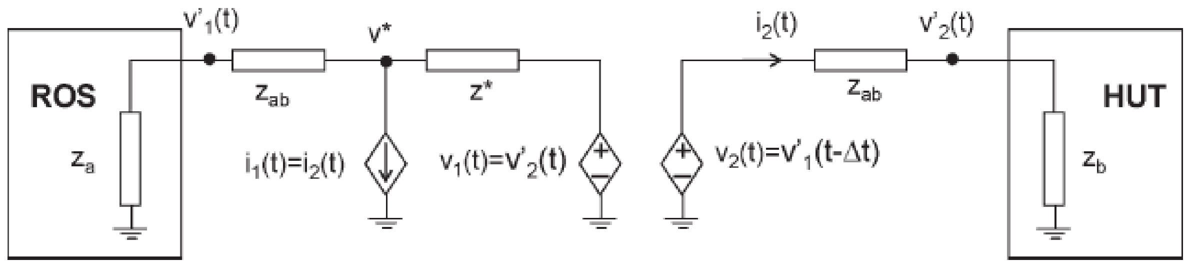


Figure 7. Co-simulation using DIM Interface algorithm model [3].

The open loop transfer function of the system is shown in Equation (4).

$$G_{OL\_DIM} = \frac{z_a(z_b - z^*)}{(z_b + z_{ab})(z_a + z_{ab} + z^*)} e^{-s\Delta t} \tag{4}$$

From Equation (4), it is noticeable that when z_b is equal to z^*, the magnitude of G_{OL_DIM} becomes zero, which assures the stability of the system and the not propagation of the error on the next time step. In [3], it is stated that to accomplish the equality of the impedances to reach a magnitude of zero is not an easy task, since normally z^* is implemented as HIL. Even though, it might be a slight difference between the impendences, the method provides good stability and accuracy.

2.3 Co-simulation Techniques

A distributed simulation is a group of components constituting a simulation system interacting over a network that can be either local or distant [28]. The distributed simulations have different approaches depending on the application it is created for. Based on [29] three main modes of Distribution Simulation can be implemented to speed up the simulations, to link the several simulations, and in some cases also to improve the simulations reusability.

Mode A implements a simulation (that might be implemented in a single computer) that is subdivided into different models. These subdivided simulations interact with each other via communications network. In Mode B, several simulations are linked by the communications networks as in Mode A, but in this case the models can exceed the capability of a single computer. The simulations, hence, can be reused by been connected to other simulations, reducing the cost of developing. Finally, the simulations of Mode C are sequentially run (one at a time) in parallel multiple computers. The simulations are coordinated by an experimentation manager via a communication protocol [29].

The time of a computer set may differ, due to the different count frequencies in the hardware clocks. Thus, the main issue regarding the distributed simulation is the synchronization of software running processes on the different computers, intended as the simulation time across the computers involved. For this purpose, many techniques to synchronize the co-simulation have been studied in the recent years. The main synchronization protocols are presented in the following sub-sections.

2.3.1 Synchronization Protocols

2.3.1.1 Network Time Protocol (NTP)

NTP stands for Network Time Protocol which is an internet protocol used to synchronize the clocks of computers to a time reference. The NTP subnet architecture is hierarchical by stratum and the subnet must be reliable and survivable in any condition, so this means that NTP requires redundant time servers [30].

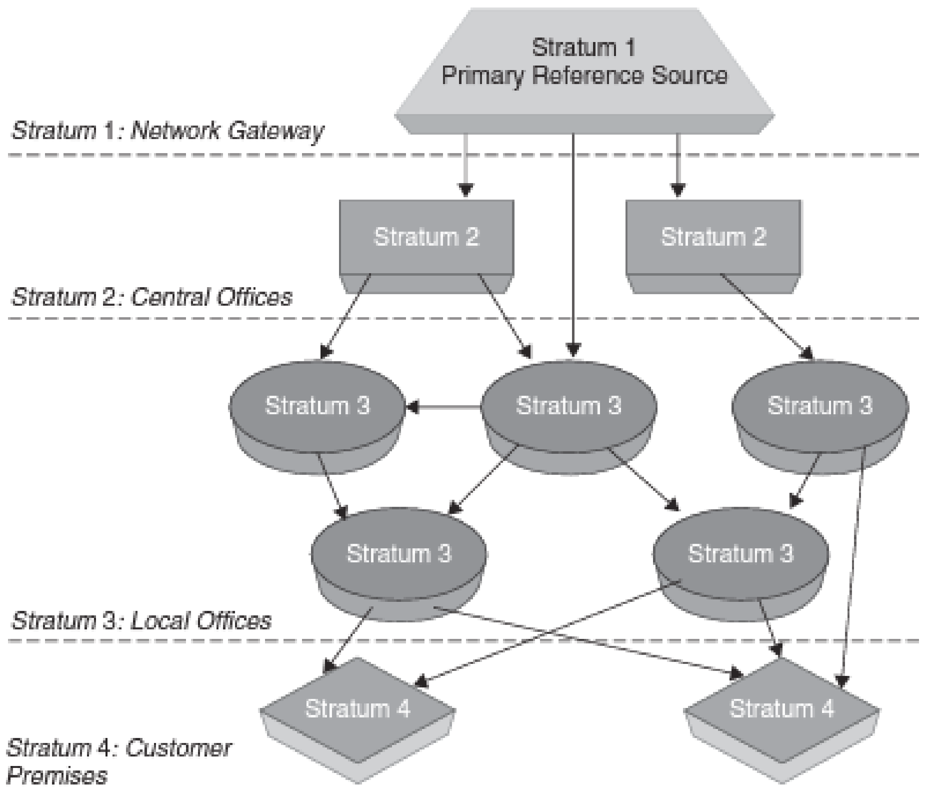


Figure 8. NTP protocol structure [31].

Based on the stratum architecture (Figure 8), the synchronization flows from the primary servers to the secondary and going on successively towards the lowest stratum [30]. The primary time server is directly synchronized to a primary reference (which at the same is synchronized to national standards by wire or radio) [32]. NTP uses Coordinated Universal Time (UTC) reference time to define the system real time [31]. The secondary servers (stratum 2) are the local-net hosts that distribute time via NTP to the remaining local-net hosts [32].

NTP has three principal modes of operation. The first one is client/server model, where the received messages by the client allows it to determine the server time with respect to local time and adjust the local clock accordingly [32].This paradigm is stateless and the servers does not need prior configuration [31].

The second operation mode is symmetric active/passive. This mode is intended for clique architectures of low stratum peers that operates as mutual backups for each other. Each peer normally works with one or more clock references or a subnet of primary and secondary servers that are known to be reliable [30].

Finally, the third mode is the broadcast intended for few servers and a large client population. The working principle is a broadcast server that continuously generates messages to the configured clients. The clients normally respond to the first messages and then each client polls the server in a client/server mode, using the burst feature. A volley of several exchanges is performed in an interval of approximately 16 s. When the volley is over, the clients set the clock and computes the offset between the broadcast time and the local time. Once the offset is calculated, the server continuous as before but the clients no longer send messages [30].

Regarding the network layer protocol on NTP; IP provides data integrity. Instead, on the transport layer protocol, NTP uses UDP over packet switched networks to synchronize clocks between one-or several-time servers and clients, and at the same time provide the checksums on the sent messages [32].

The accuracies achievable by NTP depend strongly on the local-clock hardware precision, the rigorous on device control and the encountered process latencies [32].

One of the main factors that can drive to frequency fluctuations is the ambient temperature that affects the circuit components. Different algorithms in NPT protocol are implemented to correct the frequency. For example, the discipline algorithm corrects the frequency once every second reaching to an accuracy of 100 µs, which in general is considered good timekeeping for local-area-networks (LANs). Better accuracy can be reached by using different techniques related with the clock adjustment in the kernel [30].

2.3.1.2 Simple Network Time Protocol (SNTP)

SNTP stands for simple network time protocol which is basically the NPT but does not implement internal algorithms in some servers [31]. The SNTP is intended for systems with low-end workstations that for any reason cannot justify running the full NTP implementation and do not have clients on their own [30].

SNTP has the same on-wire packet format and protocol but might lack the full suite NTP mitigation and grooming algorithms. For instance, either the primary servers and clients can run SNTP because the primary servers do not accept synchronization and the clients don't provide synchronization. Instead, when regarding the secondary server, since both accept and provide synchronization, it must conform the full NTP [30].

As specified in the NTP section, NTP clients communicate with several NTP servers to determine the correct time and also using the algorithm correction. SNTP instead, uses only one clock source as reference and normally adjust the local clock in one step [33].

Just as NTP, SNTP uses the Server/Client model. The process is initiated by the SNTP client which sends a SNTP request to the SNTP server. The SNTP server replies with a message containing the local time, then the client calculates the offset between the server and itself and adjust the local clock [33].

A study performed by Ussoli and Prytz [33] determined that it is possible to reach an accuracy of milliseconds when a sufficient timestamping accuracy and some intelligent offset calculations are implemented together. Moreover, it is specified also the necessity of using filtering to make smooth clock updates [33].

2.3.1.3 Precision Time Protocol (PTP)

In order to synchronize independent clocks running in different nodes in a distributed control system, the IEEE 1588 standard specifies a protocol that performs the synchronization at a high degree of accuracy and precision over the communication network, from a grandmaster to a slave, comprising transparent clocks (TCs) and boundary clocks (BCs) [34]. In IEEE 1588, a clock is a node implementing PTP. [35]

In an ethernet based network, the synchronization packets must share the network bandwidth with non-synchronization packets, which causes packet delay variations. For this reason, the PTP standard recommends the usage of transparent and boundary clocks to measure and correct the time the packets spend in the queue buffer of output ports. The previous solution is recommended on new installations, but regarding the costs, when dealing with existing networks, other solutions are stated; like traffic design, priority tagging of synchronization traffic, etc. [36]

PTP is based on a master-slave timing distribution among the node clocks in the system. To define the state or the function of each clock port, the best master clock algorithm (BMCA) is used, where the master-slave hierarchy is defined as well as the best master source to be synchronize to. The BMCA will avoid creating timing loop through valid network paths, but in case of device or network failure it might have to re-stablish the hierarchy and reselect a master.

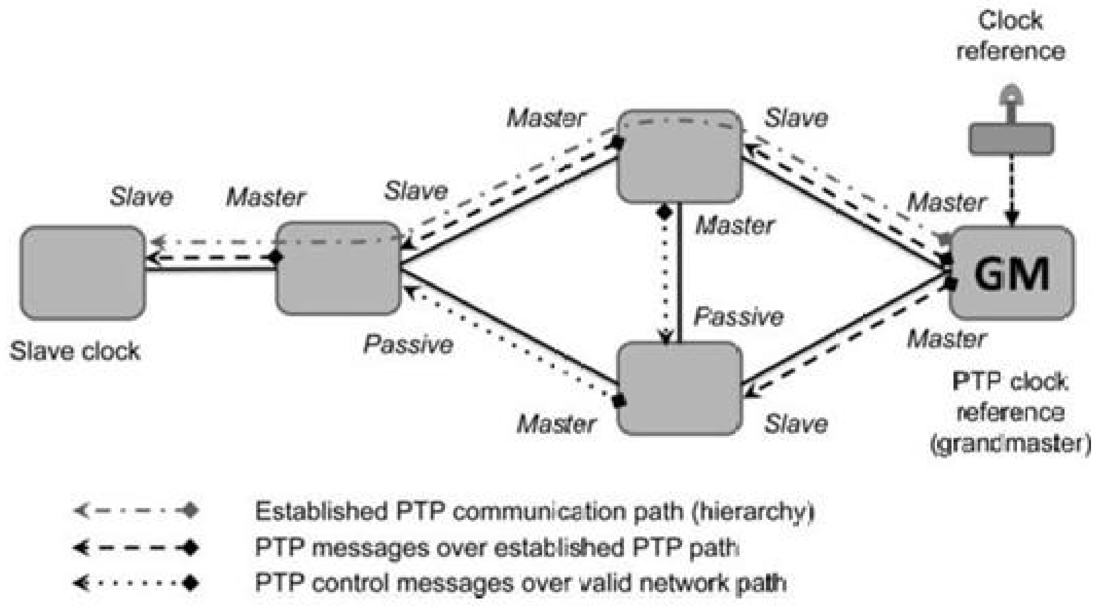

Clock reference
Master
Slave
Slave
Master
Slave
Master
Master
GM
Passive
Passive
Master
Slave clock
Master
Slave
PTP clock reference (grandmaster)
Established PTP communication path (hierarchy)
PTP messages over established PTP path
PTP control messages over valid network path

Figure 9. General scheme of the master-slave configuration ports in PTP [35].

Figure 9 shows a general scheme of the node port configuration in a system using PTP. A master port for a clock sends the packets (in the form of PTP messages) to a slave port in another clock. The slave ports allow the clock to be synchronized with the Grandmaster (GM) clock, which is the root of the hierarchy. Other than slave and master ports, a third port state exists, which is the passive port. Passive ports prevent timing loops or conflicting master ports in the same hierarchy. [35]

The BC is normally an Ethernet switch (which handles IEEE-1588 packets) that has multiple PTP ports in a domain and runs BMCA [35]. It may serve as time source, be a master clock and may be synchronized to another clock or even be a slave clock. It acts much as an ordinary clock in an isolated subnet (becomes the master clock). Boundary clocks only handles PTP packets while the standard Ethernet switches or routers handle the rest of network traffic [34]."Network engineering should ensure that all clocks participating in the PTP time distribution are set to the same domain(s)". [34, pp. 12-17].

The TC is as well an Ethernet switch (which handles IEEE-1588 packets) that measures the time taken for a PTP event message to arrive. TC provides this time arrival message measurement to clocks, in a correction field, and provides the original timestamp to the slave nodes, to allow them to determine the delay. [34] The transparent clocks in general forwards the information but does not process it, thus does not run BMCA. TC ports has no specific state and does not participate establishing PTP hierarchy [35].

The protocol appliances are focused on the cell/Area zone level based on the OSI reference model. The IEEE 1588-2002 origins contemplates operation over control local area (LANS), excluding the wide area networks (WANs) and administration free operation (being this the BMCA role). Furthermore, it is designed for minimal resource requirement in devices supporting PTP, thus clock filtering or averaging might not be necessary. [35]

Regarding the main applications intended with IEEE 1588 are included: data acquisition, military instrumentation, industrial robots and high-speed printers [35]. The protocol targets to sub-microsecond accuracy by having the master clock sending multicast synchronization message frames containing time stamps [31]. In order to get the sub-microseconds accuracy, the configuration of two devices with different parameter settings must be minimum which recalls the device profile [37] defined by the IEEE 1588 as “Profile: The set of allowed Precision Time Protocol (PTP) features applicable to a device”. [34]

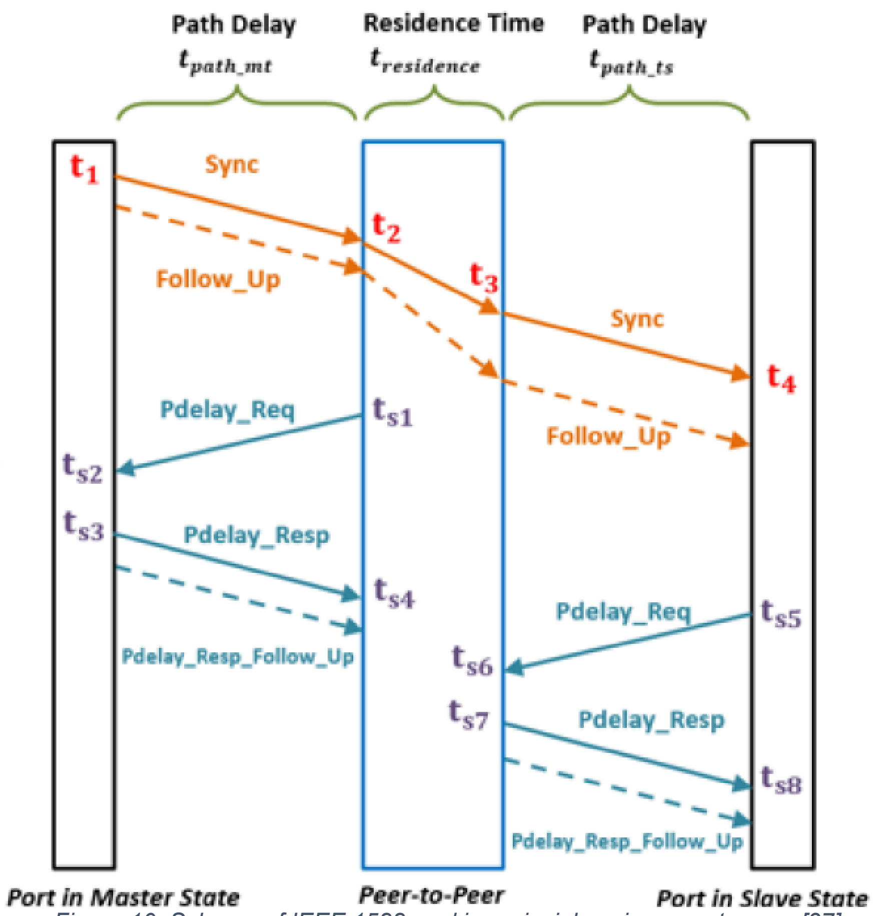


Figure 10. Scheme of IEEE 1588 working principle using peer-to-peer [37].

Figure 10 shows the messages sent from the master clock to the TC and those sent from the TC to the slave clock. In the synchronization process the Master Clock sends 2 messages: first sends Sync and after Follow_up messages. Sync messages contain an estimate of the sending time. When Sync is received by the slave, the received time is store. The Follow_up time message contains the precise sending time (measured as close as possible to the physical layer of the network) of the sync message. When this message is received by the slave clock, it is used for the calculations rather than the sending time contained in the sync message [31].

Since at the begging the TC works as a slave clock and then as a master, the Pdela_Req is a message sent from the TC/BC but also from the slave clock, after receiving the Sync message. The time the messages spend on the TC/BC are called residence time. When the master receives the Pdelay_Req, it takes note of the receipt time and responds with the Pdelay_Resp message that contains the receipt time of the Pdelay_req message [31].

Han & Crossley Conference paper [37] states the offset based on which the slave corrects its local clock (using the nomenclature showed in Figure 10). The offset calculation is shown in equations (5) and (6).

$$t_{offset} = t_4 - t_1 - t_{propagation} \tag{5}$$

$$t_{propagation} = t_{link\_mt} + t_{link\_ts} + t_{residence} \tag{6}$$

Where:

$$t_{residence} = t_3 - t_2 \tag{7}$$

$$t_{link\_mt} = \frac{(t_{s4} - t_{s1}) - (t_{s3} - t_{s2})}{2} \tag{8}$$

$$t_{link\_ts} = \frac{(t_{s8} - t_{s5}) - (t_{s7} - t_{s6})}{2} \tag{9}$$

Many analyses take into account factors that can potentially affect the overall synchronization accuracy. When considering real time substations and the traffic through the network, the study developed by Han and Crossley [37] concluded that when testing diverse network topologies, using highly redundant substations; and generating the traffic (Sample Value and Generic Object Oriented Substations Events), PTP is able to maintain the accuracy without being affected by the traffic. Thus the 1 µs can be obtained with commercial devices.

When interconnecting two or more nodes for data exchanging information, the IEEE Standard specifies that if the clocks (switches) “also support another protocol, such as Distributed Network Protocol Version 3 (DNP3) or Modbus, they may also support the mapping of the data available from either the clock datasets or TC datasets to provide that data via that supported protocol.” [34]

2.3.1.4 IRIG Protocol

The Inter-Range Instrumentation Group (IRIG) is a Rockwell Automation product that provides a synchronization accuracy range of 1 to 10 microseconds. In order to achieve this accuracy, the protocol implements the 1756-TIME module and coaxial wiring to reach synchronization through large geographical areas [38].

The target appliances are related to the military area, aerospace and power utility instrumentation [38].

The main disadvantages of this protocol are related with the additional expenses of the specialized hardware that must be added, plus the expenses related with the licenses. Another disadvantage is the increased time skew due to the required added physical infrastructure. Table 1 summarizes the protocol characteristics mentioned in this section.

Table 1. Summary table of the synchronization protocols.

Synchronization	Accuracy	physical protocol	Transport protocol layer	Hardware needed	Appliances	Topology	How it works	Drawbacks
Network Time Protocol (NTP)	10 ms on internet paths. 1 ms in LAN.	ETHERNET	NTP uses UDP over packet switched networks.	NTP server, radio or atomic clock.	Manufacturing Zone (level 3) up through the Enterprise Network.	Server-Client	Primary time server directly synchronized to a primary reference source. NTP generally uses time source like radio or atomic clock attached to main time server, then NTP server distributes the time across the network.	Asymmetries on the network can cause bigger delay transmission. NTP enabled device will not synchronize to a device whose time is significantly different than others.
Simple Network Time Protocol (SNTP)	Milliseconds or tens of milliseconds range. Depends also on the network traffic.	ETHERNET		A SNTP server to use as a time reference	Small networks with few or nule network traffic.	Server-Client	The process is initiated by the SNTP client which sends a SNTP request to the SNTP server. The SNTP server replies with a message containing the local time server, calculate the offset between the server and itself and adjust the local clock (the clock of the SNTP client).	SNTP relies on software timestamps where there may be considerable jitter due to operating systems scheduling. SNTP can only perform the synchronization every 16 s or slower.
Precision Time Protocol	10 to 100 ns achievable.	ETHERNET	IP	PTP grandmaster: GPS or CDMA	PTP is recommended to be used at the cell/Area zone level.	Master-Slave	The clocks communicate with each other over a communication network. The protocol generates a master-slave relationship among the clocks in the system by determining which of the possible sources has better accuracy. All clocks derive their time from a clock known as the grandmaster clock.	PTP is not routable
IRIG protocol	1-10 microseconds	COAXIAL. It also uses ETHERNET		1756-TIME module	Use across large geographic areas (e.g. Military, aerospace)	Rockwell Automation product		The expenses of additional HW and the increased time skew due to the added physical infrastructure required.

2.3.2 Internet Suite Communication Protocols

2.3.2.1 Request Response Approach

2.3.2.1.1 TCP/IP

In a Local Area Network every link interface has a unique address, some ways to communicate through the link interface, as well as network interface, are the usage of serial port or an Ethernet card. The network layer ensures the data transfer between two remote computers within a particular Wide Area Network (WAN) [39].

TCP and UDP belong to the transportation layer. TCP transfers data between two application running on individual computers on the internet. The client encapsulates the TCP segment into an IP datagram, which has a source IP address the "Client" and its destination is the "Server" IP address. In the case of UDP, UDP datagrams are also enveloped in an IP datagram containing, analogous to TCP, the source port and the destination port [39].

TCP/IP is a connection-oriented service, which means that a connection is first stablished between the parts (Handshaking dialogues in 4 steps) and the destination confirms the data receiving. The maximum acknowledgment time is of 500 ms. If any TCP segment gets lost, the destination asks for the retransmission of the lost data. Full duplex circuit stands for data simultaneously transferred in both directions independently [39].

2.3.2.1.2 UDP/IP

Concerning UDP, it is a connectionless-oriented service where handshaking dialogues are not performed, so this protocol does not warrantee the delivery of the messages. Therefore, the user's program is subject to unreliability of the underlying network [39].

In order to select which protocol to use, it is important to determine the type of the application intended to deploy. UDP might be an optimal protocol when in the application is preferable to drop some packages but deliver on time. Instead in TCP, will always assure the delivery of packages even because of the acknowledgment mechanism that might result in retransmission. Table 2 confronts the UDP/IP and TCP/IP characteristics.

Table 2. Summary table of Request response protocols

Characteristics	Transport Layer Protocols	
	TCP/IP	UDP/IP
Transmission time	Maximum acknowledgment time: 500 ms	Non acknowledgment
Hardware needed	In WANs need at least one router between 2 computers. The connection between two neighbouring routes on the link layer is always direct.	
Topology	Connection oriented service. Full duplex circuit (data simultaneously transferred in both directions independently.	Connectionless oriented
Data reception confirmation	The destination confirms the data received. If any TCP segment gets lost, the destination asks for the retransmission of the lost data.	No confirmation is given for the received data.
Handshaking dialogues	four step hand shaking.	Handshaking dialogues are not performed. This protocol does not warrantee the delivery of the messages.

2.3.2.2 *Publish Subscribe Approach*

2.3.2.2.1 Internet of things (IoT)

Performing the synchronization via the network using HTTP REST requests is not adequate to use as a synchronization method because incurs on increasing the synchronization time inaccuracy. The first reason is because the simulators might get connected to different servers that defer on the exact current time. Another reason is the presence of a delay between the client request (in this case the simulators) and the response of the servers. The consecutive response would be provided in a different time than the requested one, which would increase the inaccuracy on the distributed simulation synchronization.

The synchronization can be performed, as well, using a different protocol: MQTT. In this case, the IoT cloud server or message broker publishes the current time to the IoT device. The encountered disadvantage is aligned with the one found in the REST method. The time received from the broker is not accounting for forward and reverse one-way-delays (OWDs) of the packets carrying the timestamps and any response message. [40]

The evident problem is that every OWD contributes to synchronization inaccuracy at the device. Despite of the benefits of a wireless synchronization system and the fact this method does not need additional hardware, the accuracy is in the range of seconds, which depending on the application type might not be a suitable accuracy. [40]

2.3.3 Proprietary Protocols

2.3.3.1 SCADA

The SCADA (stands for Supervisory Control and Data Acquisition) is not precisely a communication protocol, it actually is an acquisition system to be access in most of the cases remotely.

"A SCADA system consists of a number of remote terminal units (RTUs) collecting data and sending that data back to a master station via a communication system. The master station displays the acquired data and allows the operator to perform remote control tasks" [41, p. 4]. The connection of the SCADA network to the LAN allows anyone within the company (with the software to access and the credentials) to get into the database. Open source software exists like: Citec and WonderWare to implement SCADA systems [41].

2.3.3.2 MODBUS

Modbus is a commercial communication protocol that was created initially to perform communication between Gould Modicon (now Schneider) programmable controllers. Nowadays is broadly used in automation industry to communicate electronic devices between each other.

The Modbus Organization manual [42] specifies that the transmission time depends on the controller type hence varies between 0.5 ms to 7 ms, depending also on the amount of data been transmitted. Another factor to take into account is the distance of the links with delays within sub-milliseconds to several seconds [42].

Two main modes can be implemented in the transport layer. The first one is TCP/IP, which is based on Client/Server communication model. In TCP/IP mode, two main architectures are defined on the specifications of the protocol [42].

The first architecture is when all the devices are connected to a TCP/IP network. The second architecture is the Interconnection of the devices performed through a “bridge” (through router or gateway). This bridge performs the interconnection between the TCP/IP network and a serial line sub-network that allows the Serial line connections between Client and Servers. Figure 11 illustrates the two architecture configurations.

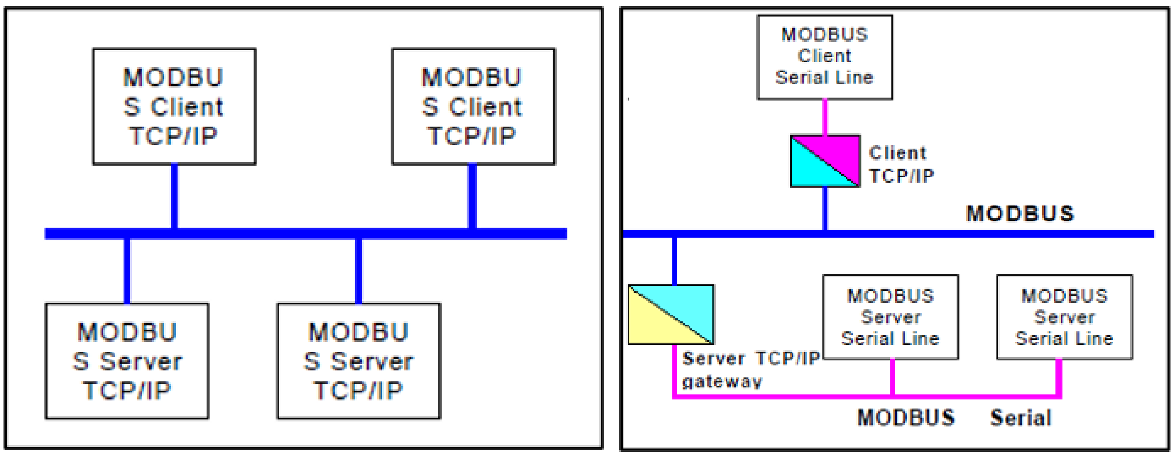


Figure 11. i) TCP/IP network architecture. ii) Serial Subnetwork of servers interconnected though a "bridge" [42].

The second implementation mode is over Serial Line Systems. In this mode, Client/Server is the model communication for the application layer. On the other hand, Master/Slaves is the model application in the data link layer [42].

Many different physical interfaces can be used; the ones specified in the manual are RS485 and RS232 [43]. RS485 needs line termination near each of the ends of the bus to minimize reflections: 150 ohm or 1nF capacitor. In the case of RS232, no termination is needed but it is recommended to be used in short lengths (less than 20 m).

Only one device, the master, can start the queries. The slaves respond by supplying the requested data to the master. Two transmission modes can be implemented that conditions the number of bits transmitted in each sent package: RTU mode (Remote Terminal Unit) and ASCII transmission (American Standard Code for Information Interchange).

2.3.3.3 IEC 60870-5-101/104

The IEC 60870-5-101 is a protocol suitable for various network configuration including point-to-point, point-to-multipoint, etc. At the link layer provides the choice to use balance (communication limited to point to point links) or unbalance communication (suitable for multidrop). However, the balanced mode of operation is allowed only over the full-duplex communication [44], [45].

Under IEC 60870-5-101, only point-to-point (two station) links can be balanced. Multipoint links must be unbalanced. It is important to state that in the balanced communication, collision problems can happen, which means that the two stations can transmit simultaneously (the master and the outstation) if a full-duplex channel is used. In the case of the multipoint configuration the master communicates on parallel to all outstation. Since all the outstations share the communication channel, only one may transmit at a time [45].

When the master station is transmitting a message, if a confirmation is not received from the outstation within a configured time-out period, it will re-transmit the message up until a configured number of retries [45].

In IEC 60870-5 standard there are two different methods of transporting messages. implemented in two different but closely related protocols. The first one is IEC 60870-5-101 (or T101) which provides bit-serial communications over low-bandwidth communications channels. Therefore, in the physical layer, serial interfaces are stated to perform the communication, as RS232 and RS485. The addressing is performed both at the link and at the application level [45].

The second method was defined much more recently with the release of the IEC 608705-104 (or T104). IEC 60870-5-104 is an extension of IEC 60870-5-101 protocol with changes in the physical, transport and network layers to make the protocol suitable for the full network access. This means that the standard uses TCP/IP interface to be able to connect whether the connection between devices is performed within a LAN or a WAN (routers in different facilities) [44].

2.3.3.4 AURORA Protocol

Based on the Aurora 8B/10B Protocol Specification (2014), Aurora is an independent protocol that can be used to transport standard protocols as the Ethernet and TCP/IP. This light weighed protocol can be used to move data point-to-point across one or more serial speed lanes.

The protocol contains in its physical layer the electrical levels, the clock encoding and the symbol coding. Each Aurora 8B/10B lane is a full-duplex serial data connection. Figure 12 shows the structure the protocol uses where is stated that the devices that communicate across the channel are called channel partners [46].

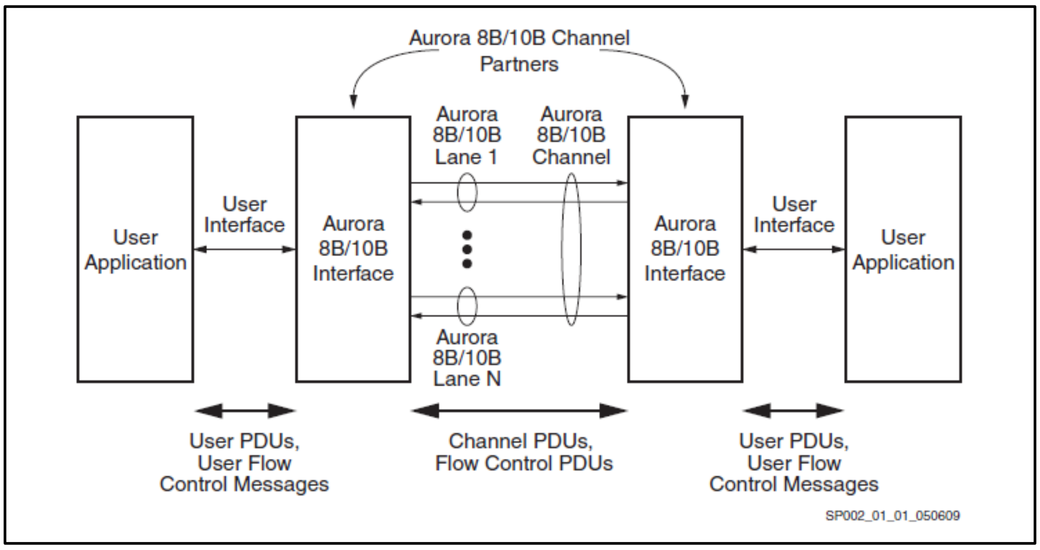


Figure 12. Description [46].

Aurora provides a compensating mechanism for clock rate differences between transmitter and receiver. This mechanism, called clock compensation, can accommodate up to a+-100 pulses per minute (ppm) clock rate differential between the transmitter and the receiver. The Aurora protocol implements clock compensation periodically inserting clock compensation sequences into idle patterns or user data [46].

2.3.3.5 DNP3

DNP3 is an open communication protocol that arises from the frame format specifications in IEC 60870. This protocol provides interoperability between devices coming from different vendors and does not need protocol translation. The supported topologies are peer-peer, Master-Slave with multi-slave, and multiple master, multidrop and hierarchical with data concentrators [45] as shown in Figure 13.

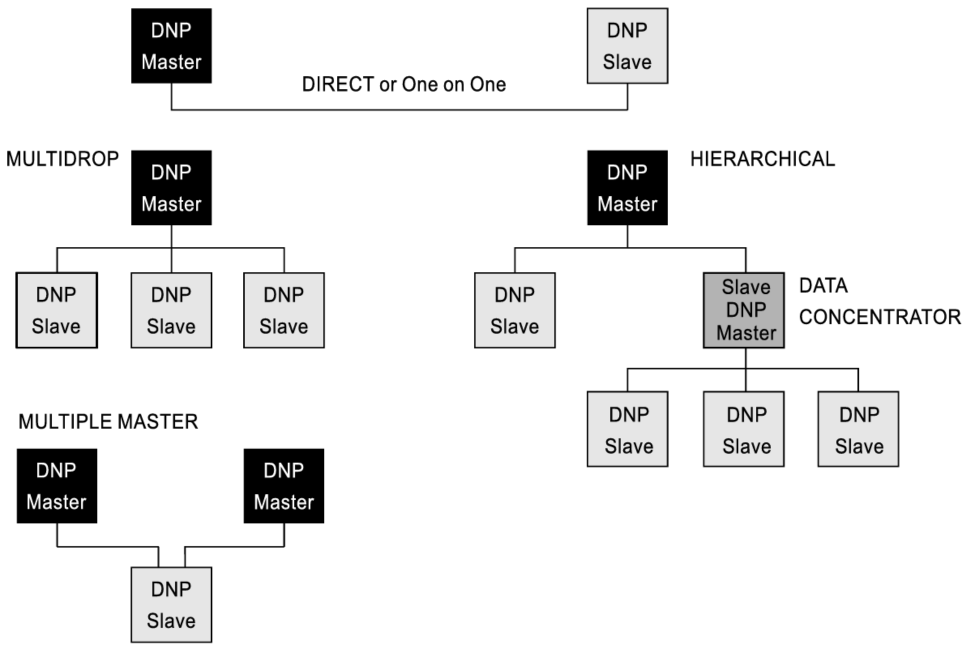


Figure 13. DNP3 topologies [45].

Regarding the physical layer, DNP3 can communicate by RS-232 C, CCIT V.24 for DTE-DCE signalling. The data transmission possesses 8 data bit, one of those is a start bit and a stop bit (bit serial asynchronous communication) [45].

Table 3. Proprietary communication protocols summary table.

Protocols		Transmission time	physical protocol	Transport protocol layer	Hardware needed	Topology	How it works
SCADA	The SCADA is an acquisition system to be access in most of the cases remotely. The master station displays the acquired data and allows the operator to perform remote control tasks. Open source software exists like: Citec and WonderWare to implement SCADA systems.						
Modbus	**TCP/IP**	Depending on the controller type varies between 0.5 ms to 7 ms. It depends also on the amount of data been transmitted and the distance of the links.	Ethernet cable, or serial cable	TCP/IP (can be implemented together with serial)	The communication between client and server MODBUS module requires TCP connection management module. The user application can manage the TCP connection itself.	Client/Server. Ethernet TCP/IP OR Interconnection devices like bridge, routers for interconnection between the TCP/IP network and a serial line sub-network.	
	Serial		Serial cable	RS485 OR RS232	RS485 (line termination of 150 ohm or 1nF capacitor) OR RS232 (no termination needed). To use on less than 20 m cable length.	Client/Server (Application layer). Master/Slaves (Data Link Layer)	Only the master can start the queries. The slaves respond by supplying the requested data to the master. Two transmission modes: **1.** ASCII transmission Mode **2**. RTU mode
Aurora		If Multilane, depending on the baud rate, but it is in the order of nanoseconds. In the single lane is in the order of picoseconds.	Can be implemented using copper or optical fiber	Full-duplex serial data connection with Ethernet TCP/IP 2 or optical fiber.	When implementing optical fiber, converting box is needed.	An Aurora 8B/10B channel consists of one or more Aurora 8B/10B lanes. Each lane is full duplex serial data connection. The devices that communicate across the channel are called data partners.	
DNP3			RS-232 C, CCIT V.24	DNP3 adds pseudo transport for transmission of larger data blocks.		Peer-peer, Master-Slave with multi-slave, and multiple master, multidrop and hierarchical with data concentrators.	The data transmission possesses 8 data bit, one start bit and one stop bit, bit serial asynchronous

2.4 Ad-hoc Interconnection

The ad-hoc interconnection is an alternative of the standardized simulation frameworks, where the simulators are interface only between them and possibly some adaptations have to be done to perform the interconnection. [16] states that ad-hoc interconnection is a very used practice since most proprietary simulation tool developers does not provide interconnection interfaces between simulation frameworks. Also, it is important that this type of connection assures conciseness and efficiency.

3 State of Art Solution

In the last decade, there has been a recent interest in accurate simulation of power electrical systems (e.g. EMTs, AC grids, ICT). The computing technology advancements, such as parallel computing, using co-simulation and digital signal processing techniques [47], has addressed research to different frameworks to perform rapid prototyping of power electric systems. Many research documents focus on the types of simulators used or developed and the frameworks implemented to perform co-simulations. These frameworks exploit different types of simulators (stepped time simulator, event driven simulator or real-time simulators) to implement co-simulation allowing to simulate complex and multi-rated systems.

For instance, in [48], it is stated that Electromagnetic transient (EMT) software normally simulates large AC and modular multilevel converters (MMC) of multiterminal systems (MTDC) at the same time step rate, which means a waste of computational resources because the AC system does not need such a fast time step. Therefore, it is proposed to separate the simulation in sub-systems, the fast system MMC MTDC and the slow AC system. To interface the systems, a RM algorithm is developed as the coordinator, which is responsible for the interactive signals exchange between simulators. The communication network between the systems is based on shared memory techniques.

A different approach to develop a multi-rated co-simulation is performed in [4]. In this case, it is interfaced a multidomain (phasor domain and continuous time) transmission line model by partitioning the network. The AC grids contained the shifted frequency phasor, composing the first subsystem that possesses the larger timestep (up to 500 microseconds) and the wind farms contained in the EMT is the second subsystem which can be simulated between 2-50 microseconds.

In [49] instead, the division of the fast and slow models was performed by exploiting a delayed signal and zero crossing signal (implemented by a switching system). A based real time co-simulation is performed by using FPGA+DSP+PC, where PC and DSP are distribution real time simulators. The control part of the inverter is simulated in the DSP at 2 microseconds (which is considered the slow rate control sub-model) and as an alternative the real time simulation platform that works at 500 ns is the high speed subsystem (fast inverter sub-model). In this model, the simulation clock is mainly controlled by the FPGA.

Another proposed technique to improve the computational efficiency when multi-rated simulation is needed, regarding the different domain issue, is an automatic solver for the selection of the time step algorithm embedded with Ordinary Differential Equations (ODEs) to determine the time step proposed in [50]

In [5] a co-simulation between RTDS and FPGA was performed, but using only EMT models to avoid the complexities of interface design and the consequent potential interface errors. The larger part of the system but that requires less modification was simulated on the FPGA. Instead, the RTDS was used to simulate the part that needed more modifications, by utilizing the RTDS flexible features to simulate. In this case, the FPGA side synchronizes its computations with the RTDS at each timestep through a synchronization signal. The interconnection between the systems was done through optical fiber with the communication protocol Aurora. It was concluded that this is a good approach to provide an efficient extension of a real time simulator (in this case by the use of RTDS) avoiding the interface errors in hybrid electromagnetic transient (EMT)- transient stability (TS) simulations.

When it is intended to co-simulate independently executed simulators to interface domains, middleware (so-called master algorithm) are used to coordinate the communication between simulators. When co-simulation of different simulator types is performed, the main problem that arises is the synchronization between them, since most of the times the simulation of the network is done in an event-driven simulator and the electrical grid in a stepped-time simulator.

Regarding the architecture to co-simulate power and communication systems, [1] and [51] classify the co-simulation platforms in three types. The first architecture category is the unified configuration [1], or simultaneous configuration as named in [51]. It consists on implementing communication system model in a power system simulation tool. One simulator is selected as the master and the other one is enslaved in the master component. In this model type the accuracy is sacrificed on behalf of the speed.

The second category is the non-real time distributed configuration [51]. In this model the coordination and synchronization between two different oriented simulators is entrusted to an independent component. Since this method relies on professional dedicated simulation tools, the accuracy of the hybrid system is maintained since the calculating method is mature. Studies implementing this method are being focused on how data is exchanged between simulators, time synchronization, etc. [1].

Many platforms concerning the non-real time distributed configuration have being studied. [52] performs a comparison study between High-Level Architecture standard (HLA) and MOSAIK, both are co-simulation frameworks that perform synchronization of the simulation models. The integration of a component in MOSAIK requires implementation of the component-API. HLA can synchronize a time-stepped simulator and a discrete-event one by letting the simulators choose the type of operation on each time step that goes forward. The system is based on a published-subscribe mechanism, where all the subscribed federates will receive the information, whether is needed or not. The stated drawback of HLA is that the existing simulations can be difficult to modify [7].

Another important platform is presented in [7]. EPOCHS is an agent- based framework that synchronizes electric power simulator (PSCAD/EMTDC) with communication simulators (NS2) allowing in this way to perform research regarding electromagnetic scenarios involving communications. In this architecture, the run time infrastructure (RTI) is the responsible of synchronizing and routing communications between components since it is in charge of receiving synchronization messages from the simulators. However, RTI synchronizes simulation time clocks rather than real time clock. On the other hand, the Agent HQ is the component that orchestrates the execution of each agent. When all agents have executed, Agent HQ returns control to RTI. RTI informs NS2 and power system simulator that the time step is finished.

Other broadly mentioned co-simulation platforms are Global Event-Driven Co-Simulation Framework (GECO) and INSPIRE. The first one uses a global event scheduler to interface the power system simulator and the communication network simulator, where each iteration is treated as a time tagged discrete event. Instead, INSPIRE performs stepped time-based synchronization using the HLA time management services [16].

The third category is the real time distributed configuration. In this case, both simulators must run accurately at the same speed being both synchronized to real world clock and exchange real time data through protocols. But, this technique is costly and complex, and regarding research, it is in its initial stage [1], [51].

Some studies has focus their research on creating new simulator in order to fulfil the demands of complex applications, like [12], and also to perform co-simulations by synchronizing simulators based in different models as the event-based simulators and the time stepped simulator as [7] and [16].

In many other documents, like [4] , [49] and [48] by implementing different approaches, the aim is to solve the problem of multi-rated co-simulation. But there is lack of information regarding the co-simulation using communication protocols in the same simulator or in performing the co-simulation between two or more simulators.

In [5], the co-simulation is performed interconnecting FPGA with a real time simulator as RTDS through Aurora protocol, that is a similar approach to which this thesis intends. Despite the fact of the similarities of the model stated in this thesis and the one performed by [5], this thesis aims to study the latency performing communication through the Aurora protocol and not to study the co-simulation as a possibility to extend the real time simulator, that was the main goal of [5]. Although a future scope of the Energy Center is to co-simulate large and complex electric systems as the EMT models of AC components like in [5], the simulators to be used are RT-LAB and RSCAD communicating information through the AURORA link, and exploit and evaluate the capabilities of the implemented framework.

This is why this thesis proposes the first steps to co-simulate two real-time simulators, by analysing the delay when implementing a simple model of co-simulation in the same real-time simulator and determining the delay when transmitting different amount of data through the AURORA link communication protocol.

As a second step, this thesis analyses a simple test case with a system divided in two parts. The sub-systems are co-simulated in the same real-time simulator by interconnecting them with the communication protocol Aurora using an echo link.

The experiment is planned to be extended to a broader scope, where the RTDS and RT-lab, both real time simulators, are going to be interconnected to perform co-simulations. Since both RTDS and RT-lab support the Aurora protocol and the protocol specification is open and has no implementation cost, Aurora was selected as the communication protocol to interconnect the simulators.

4 Methodology

This chapter is focused on the methodology used to develop the test cases. It is explained in a detailed manner how the test cases were implemented and the main problems that were faced. Once stated the problems, the developed solutions are explained.

4.1 Latency Calculation

As stated in the solution of the state of art, the implementation of co-simulation interfacing different simulator through platforms have been widely studied, as well as the co-simulation dividing the systems in a real-time simulator and PHIL. But, since there is a lack of information of co-simulation by dividing a system in the same real-time simulator by the use of communication protocols or using more than one simulator, the first step is to calculate the latency when transmitting and receiving messages or more signals from one subsystem to another.

The RTDS hardware is placed in the Energy Center of Politecnico di Torino. In the RTDS cubicles, the chassis are mounted. In chassis 2, the echo link implementation was performed by connecting through optical fiber ports 23 and 24 (because ports from 21 to 24 are Aurora reserved). The chassis is also connected to a workstation server, using the workstation interface cards in the chassis to be connected to a workstation NIC. In the workstation server, RTDS software and applications are installed to configure, simulate and monitoring the models.

In order to reach efficient and satisfying results, different tools were used in the implementation of the solutions as python and the RSCAD applications and its features. Once the procedure to develop the experiments is stated, the results are presented and analysed.

To have a better understanding of the system, the generated documents in the compilation and in the running of the test cases, it was very important to handle the Control system Components manual and the script reference manual. The tutorial use cases and compilation information contained in the manuals were also useful to develop the test cases.

4.1.1 Test Case Definition

The first part of this test case consisted on determining the time delay between the transmission and reception of sent data points using Aurora link. To reach this objective, the network ports 23 and 24 (of the RTDS) were connected in a closed loop using an optical fiber cable, as stated in the previous section. Both ports were configured to send and receive data as aurora link ports.

To determine if a variation on the transmission time delay from one port to another is affected by the quantity of data transmitted or the timestep rate variation, this test was performed transmitting the smallest allowable amount of data: 2 inputs from port 23 and from port 24 receiving 1 point, to the greatest amount of data: 128 outputs (port 23) and 128 inputs (port 24).

4.1.2 RTDS

As a first step, a model to measure the delay to transmit data from one point to another in the same real time simulator (RTDS) was created in the dft (draft document). Then through python, the model was replicated to create the documents needed to simulate from 2 inputs to 128. For each time step, 126 documents (for each .dft and sib. File types). In these documents the time step and the simulation time were imposed.

The time steps selected to simulate were 7, 10, 15, 20, 25, 30, 35, 40, 45, 50 microseconds. The goal was to simulated from 5 microseconds up to 50 microseconds, with a difference of 5 microseconds up to 50 microseconds, but due to the capability of the system to collect data equivalent to 64 MB, when trying to simulate under 7 microseconds the simulation crushed, so the smallest time step that was possible to simulate was 7 microseconds.

Finally, to perform the simulation, a script code was created to automatically perform the simulation for each timestep. In this way, the script automatically simulated the variables from 2 to 128 points, for an imposed constant time step.

The simulation time was calculated from the amount of point that the system allowed to simulate, since only 64 MB are available to collect data points. This procedure was performed experimentally, trying to simulate the maximum amount of points, until the script file didn't crush. Every time the simulation crushed, all .dft and .sib files had to be adjusted with the appropriate simulation time for the imposed timestep, since the amount of points simulated had to be the same, for every timestep, in order to reach consistency in the statistical analysis.

The script created automatically a CSV file (a excel file). This file permits to analyse the time delay transmitting information from one port to another for a certain imposed time step for the simulated number of variables (since one CSV files was created for each number of variables from 2 to 128, and for each time step). The document creation can be seen as a matrix composed of two variables: the time step and the transmitted variable amount, so since 10 time steps were simulated and for each timestep 126 transmitted variable documents were created, the total quantity of created CSV files is 1260.

4.2 Simple Case Study

Once the time delay to transmit a message from one port to another is known, a simple case study to determine the system behaviour was set. It was intended to implement a simple circuit co-simulated using the aurora link connecting the network ports 23 and 24 (of the RTDS) in a closed loop using an optical fiber cable, as in the configuration described in the test case definition.

4.2.1 Test Case Definition

The initial model was configured as a simple AC source, connected to two resistances. Then the circuit was divided in two parts. The first part was composed by the AC source, the first resistance (Z_A) and a remote-controlled current injector. The second part of the circuit contains a remote-controlled voltage injector, and the second resistance (Z_B). The remote-controlled current injector from the first part of the circuit was controlled by the feedback of the current in the second part of the circuit, sent through an Aurora link. In a similar way, the remote-controlled voltage was driven by the feedback of the voltage measured after the Z_A (V_A), on the first part of the circuit, and was sent also through an Aurora link.

4.2.2 Interface Algorithm for Digital Real-time Simulation

4.2.2.1 ITM Stability and Accuracy Application to RDTS Co-simulation

Before simulating on the real time simulator, it was necessary to determine under which parameters the system was stable. As a first step, it was analysed a suitable interface algorithm for a system with delays. The Ideal Transformer Model was selected as the interface algorithm to analyse the stability of the transfer function, since it is simple to deploy and possesses a high accuracy.

The theoretical analysis of the system was performed through MATLAB, to determine the stability of the system for the whole range of impedances to be used. After determining the values of impedances that assured the system stability, it was proceeded to simulate on the RTDS. Three cases were tested on the real-time simulator. The first case was the system naturally coupled, the second one as the system decoupled but without using the protocol communication to control the remote signals, and the third case was the system decoupled and by controlling the remote signals using the Aurora link.

Attention was focused on the most critical cases found on the theoretical analysis, to determine the behaviour of the real time simulator and perform a comparison of the theoretical analysis with respect to the experimental one.

5 Analysis of Numerical Stability and Accuracy

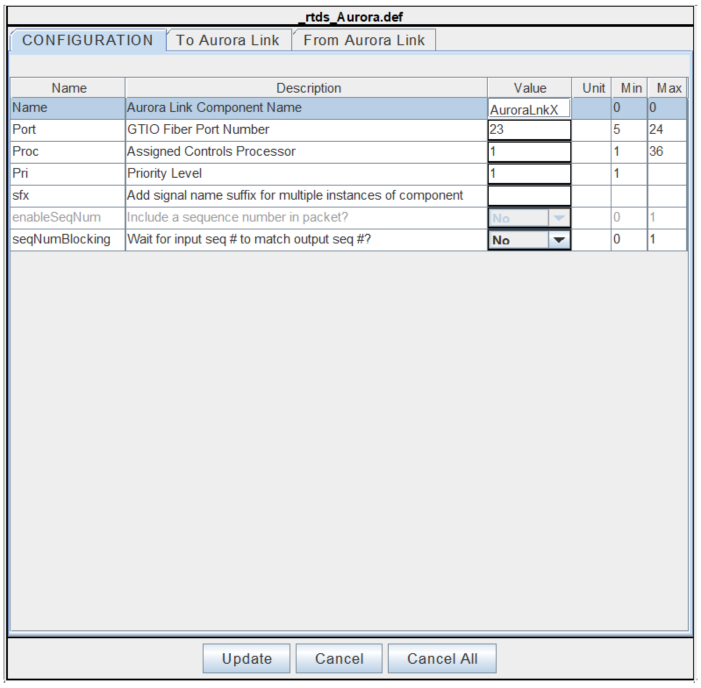
5.1 Latency Calculation

This chapter describes the procedure to perform the latency calculation when using Aurora protocol as a link to transmit data from one port of the simulator to another in the same real-time simulator (RTDS). It is also determined if the amount of data transmitted influences the time delay of the data transmission.

5.1.1 RTDS

The first step of all was to create the draft document. The Aurora Link algorithm in its configuration window allows to set the port to be use. In this case, as stated in the Methodology (Test case definition sub-section), since a loop from port 23 to 24 (Figure 14) was implemented, the configuration of the first block was port 23. After this, the setup of the To Aurora Link tab and From Aurora Link tab must be completed.

In the To Aurora Link, the minimum data amount allowing the algorithm to work is 2 variables, instead in the *From Aurora* sub-window the minimum number of variables allowable is 0. In both cases the maximum amount of data is 128 variables. To be able to transmit and receive more than 64 variables, an update must be performed and then the restart of the hardware.



Name	Description	Value	Unit	Min	Max
Name	Aurora Link Component Name	AuroraLnkX		0	0
Port	GTIO Fiber Port Number	23		5	24
Proc	Assigned Controls Processor	1		1	36
Pri	Priority Level	1		1	
sfx	Add signal name suffix for multiple instances of component				
enableSeqNum	Include a sequence number in packet?	No		0	1
seqNumBlocking	Wait for input seq # to match output seq #?	No		0	1

Figure 14. Aurora block configuration window.

The draft document was created with 128 variables. Each variable is a timer which start counter is controlled by a switch, and the outputs of the timer are labelled from D1 to D128. The timer algorithm was used because of the facility on changing the output while time increases. Figure 15 illustrates the composition of the variables.

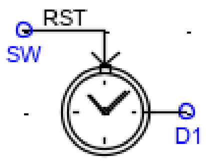
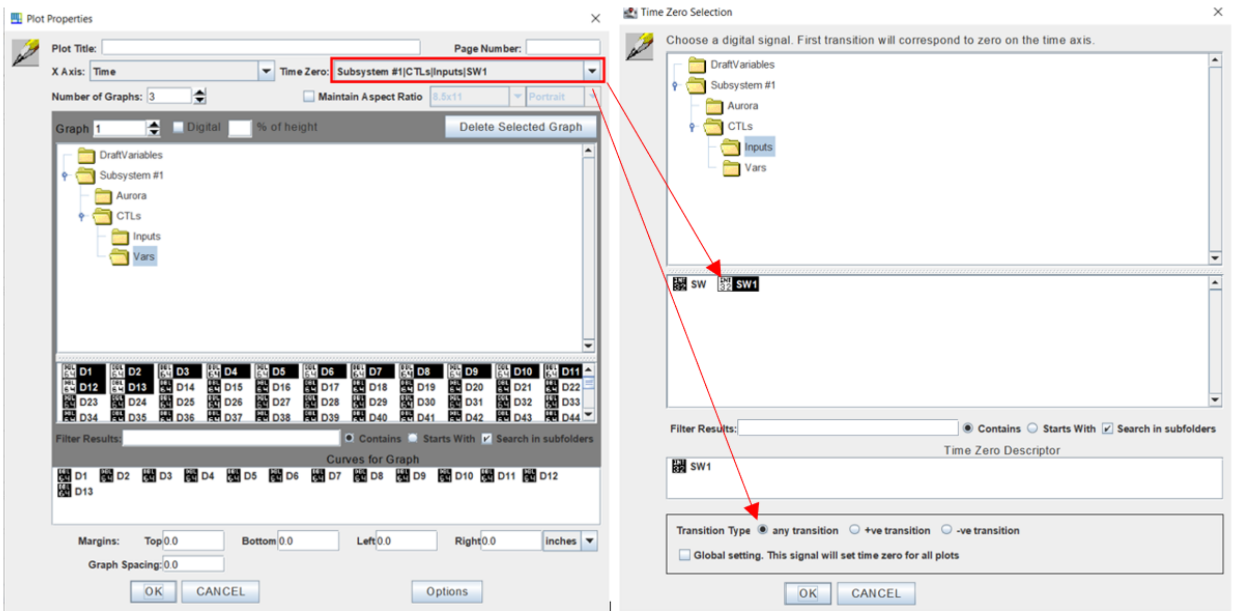


Figure 15. Timer activated by a switch.

The RunTime is able to show and store data simulation by the creation of a plot and the consecutive adding of the points in that plot. When the simulation starts, the data is not shown in the graphic until the user presses the update button and the simulation time of the *.sib* document has already been passed. After the update button was pressed and the simulation time has been passed the data appears and only after this the user is able to save the data in a .csv document.

One of the first problems found, when beginning a single timer simulation, was that the timer started the count before it was possible to update the graphic. When saving the data, the information shown at zero or close to zero was near 1.2 s, which meant that the manual user update was too slow and inefficient to collect the data.

Therefore, a system that interfaced the RunTime and the Draft document was implemented through a switch (Figure 16). The switch at the beginning was set to 1 which resets constantly the timer. When the simulation started, the switch was changed from 1 to 0 allowing the counting to start. In the Plot RunTime the zero signal was also set as the transitions of the switch (any transition) simulating that the actual starting time of the simulation was the moment when the switch changed.


Plot Properties

Plot Title:
Page Number:
X Axis: Time
Time Zero: Subsystem #1|CTLs|Inputs|SW1
Number of Graphs: 3
Maintain Aspect Ratio 8.5x11 Portrait
Graph 1 Digital % of height
Delete Selected Graph

DraftVariables
Subsystem #1
Aurora
CTLs
Inputs
Vars

D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22
D23 D24 D25 D26 D27 D28 D29 D30 D31 D32 D33
D34 D35 D36 D37 D38 D39 D40 D41 D42 D43 D44

Filter Results: [● Contains] Starts With [✓ Search in subfolders]

Curves for Graph
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
D13

Margins: Top 0.0 Bottom 0.0 Left 0.0 Right 0.0 inches
Graph Spacing: 0.0

OK CANCEL Options

Time Zero Selection

Choose a digital signal. First transition will correspond to zero on the time axis.

DraftVariables
Subsystem #1
Aurora
CTLs
Inputs
Vars

SW SW1

Filter Results: [● Contains] Starts With [✓ Search in subfolders]

Time Zero Descriptor
SW1

Transition Type [● any transition] +ve transition -ve transition
Global setting. This signal will set time zero for all plots

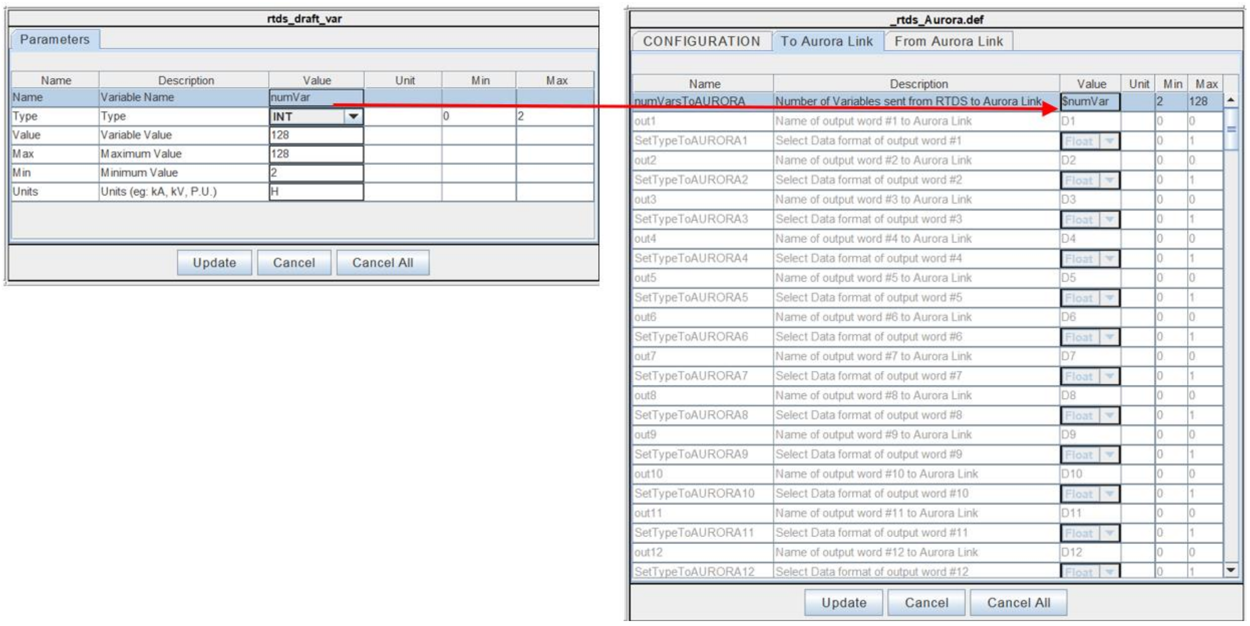
OK CANCEL

Figure 16. Configuration of the switch on the RunTime.

Since the objective of this test case was to determine if the delay of transmission of data might be affected by the number of variables or the time step, it was necessary to develop a method where the number of transmitted variables through the Aurora Link could be changed.

To implement the changing of the variables in the Draft, it was used the algorithm *rtds_draft_var* that allows to state a variable that will be passed through the *RunTime*. The name of the variable is set in the algorithm and then used in the Aurora configuration window. The *numVarstoAurora* enables the transmission of the set quantity. The same happens on the From Aurora Link sub-window.

Figure 17 shows the configuration of the *_rtds_Aurora* algorithm, where the variable defined in the *rts_draft_var* must have a "$" simbol in front.



Name	Description	Value	Unit	Min	Max
Name	Variable Name	numVar			
Type	Type	INT		0	2
Value	Variable Value	128			
Max	Maximum Value	128			
Min	Minimum Value	2			
Units	Units (eg: kA, kV, P.U.)	H			

Name	Description	Value	Unit	Min	Max
numVarsToAURORA	Number of Variables sent from RTDS to Aurora Link	$numVar		2	128
out1	Name of output word #1 to Aurora Link	D1		0	0
SetTypeToAURORA1	Select Data format of output word #1	Float		0	1
out2	Name of output word #2 to Aurora Link	D2		0	0
SetTypeToAURORA2	Select Data format of output word #2	Float		0	1
out3	Name of output word #3 to Aurora Link	D3		0	0
SetTypeToAURORA3	Select Data format of output word #3	Float		0	1
out4	Name of output word #4 to Aurora Link	D4		0	0
SetTypeToAURORA4	Select Data format of output word #4	Float		0	1
out5	Name of output word #5 to Aurora Link	D5		0	0
SetTypeToAURORA5	Select Data format of output word #5	Float		0	1
out6	Name of output word #6 to Aurora Link	D6		0	0
SetTypeToAURORA6	Select Data format of output word #6	Float		0	1
out7	Name of output word #7 to Aurora Link	D7		0	0
SetTypeToAURORA7	Select Data format of output word #7	Float		0	1
out8	Name of output word #8 to Aurora Link	D8		0	0
SetTypeToAURORA8	Select Data format of output word #8	Float		0	1
out9	Name of output word #9 to Aurora Link	D9		0	0
SetTypeToAURORA9	Select Data format of output word #9	Float		0	1
out10	Name of output word #10 to Aurora Link	D10		0	0
SetTypeToAURORA10	Select Data format of output word #10	Float		0	1
out11	Name of output word #11 to Aurora Link	D11		0	0
SetTypeToAURORA11	Select Data format of output word #11	Float		0	1
out12	Name of output word #12 to Aurora Link	D12		0	0
SetTypeToAURORA12	Select Data format of output word #12	Float		0	1

Figure 17.Number of transmitted points by Aurora as a variable

The reason why the draft was created with 128 variables is because is the maximum number of variables supported by the Aurora Link. The .dft document generation was done through a python program (APENDIX A: PYTHON CODE TO GENERATE THE .DRAFT AND .SIB DOCUMENTS) by studying and analysing the structure and pattern of the .dft document.

The program created in python generates a .dft document that contains from 2 to the number of variables that the user introduces. Two way of creating documents were implemented. The user can decide to create a document whether for each variable from 2 to the number introduced to the program or a single document by specifying the step between the creation of the documents.

For the matters of this experiment a single *.dft* document was enough to perform the simulations. Figure 18 shows the final *.dft* document used to develop the simulations.

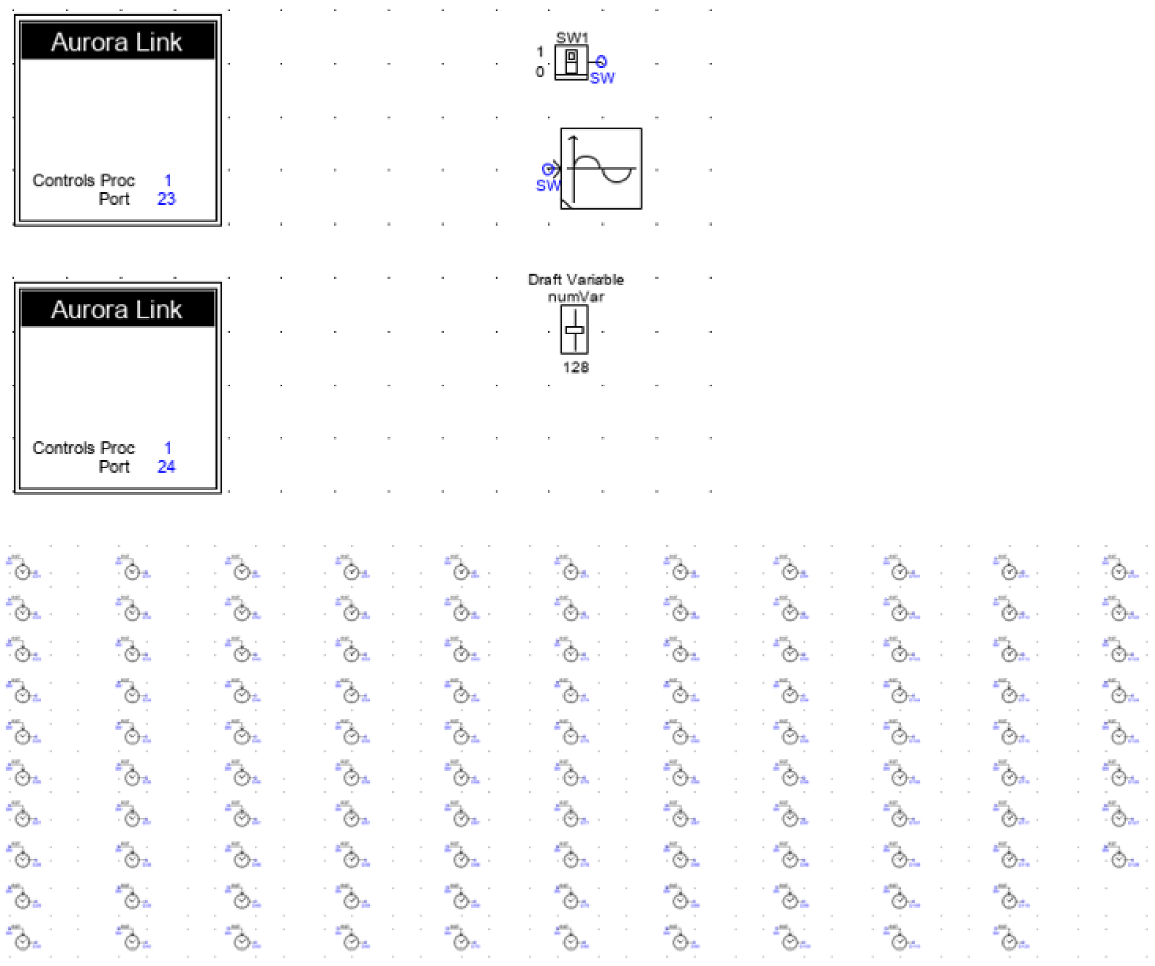
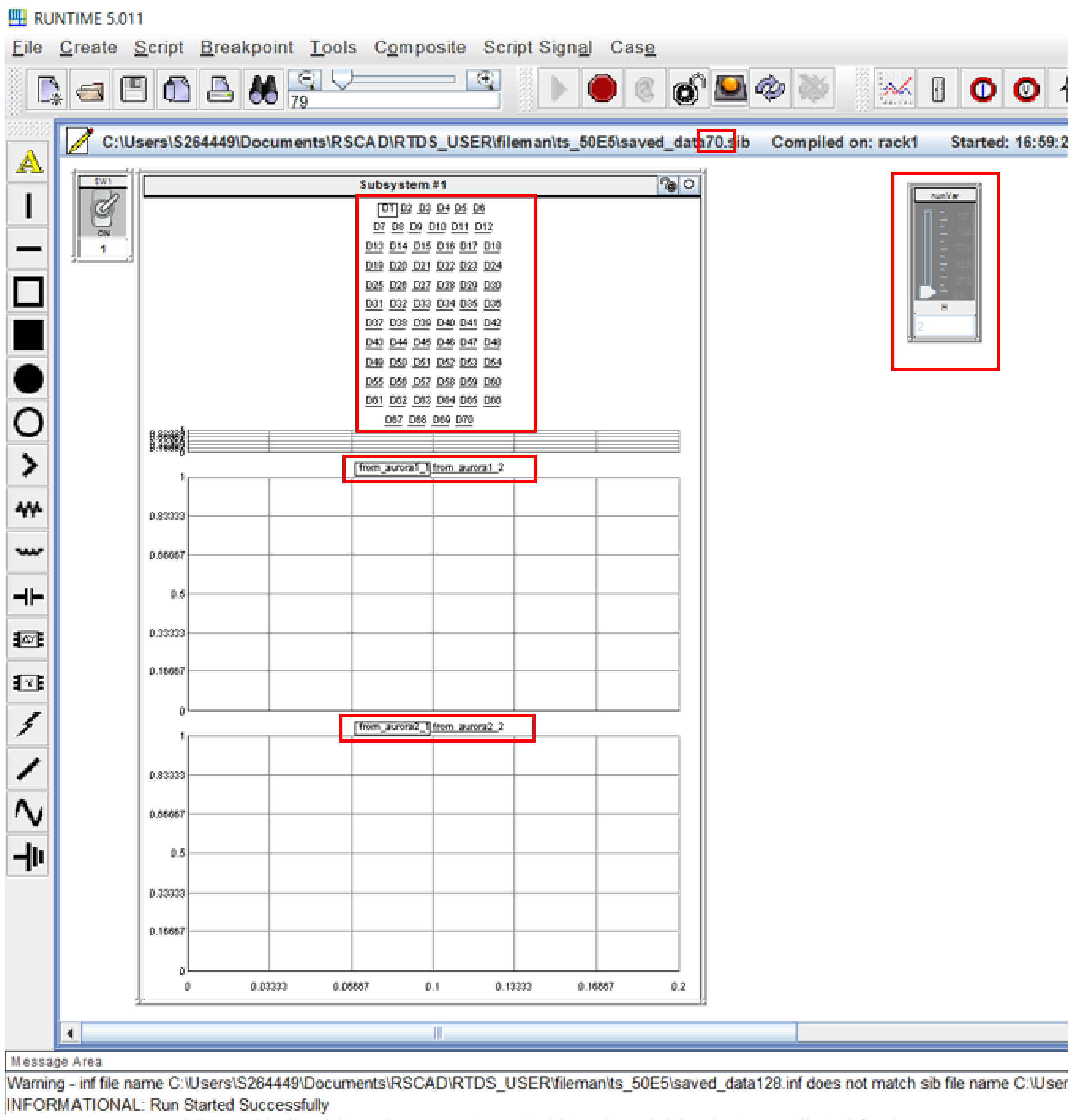


Figure 18. Draft containing 128 variables transmitted through Aurora Link.

Regarding the RunTime, when performing a single simulation, three elements must be in the document: the switch, the plot containing the labelled points to be sent and received, and a slider. The slider is the interface that allows the user to change the number of variables been sent by the Aurora Links.

In the plot, the timer outputs were labelled as *Di* (where i is a number from 1 to 128). The inputs to the Aurora Link 1 (port 23) were labelled as *from_aurora1_i*, and the inputs to Aurora Link 2 (port 24) were label as *from_aurora2_i* (where i is a number from 1 to 128). Both aurora blocks sent the outputs of the timers (*Di*) and received a labelled signal as detailed before.

One important detail to state is that each time the slider number is changed, when starting the simulation, the system recompiles the draft, and the slider freezes until the simulation is finished. When the draft recompiles for a certain number of variables, the *sib* document is also changed to the number of variables for which the draft was compilated. Therefore, the plot switches to the number of variables even if it was created to plot a greater number of variables. The graphic doesn't allow to add variables for which the draft has not been compiled for.


RUNTIME 5.011
File Create Script Breakpoint Tools Composite Script Signal Case
79
Subsystem #1
from_aurora1_1 from_aurora1_2
from_aurora2_1 from_aurora2_2
Message Area

C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\saved_data70.sib Compiled on: rack1 Started: 16:59:2

Warning - inf file name C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\saved_data128.inf does not match sib file name C:\User
INFORMATIONAL: Run Started Successfully

Figure 19. RunTime document created for 70 variables but compilated for 2.

Figure 19 shows an example of what happens when the RunTime document has been created for 70 variables, but compilated for 2. The rest of the variables that are not going to be received by the Aurora Link disappear. This means that in the case of the variables that are received by the link, if for instance, there are enabled only 2 spaces by the slider, it is traduced by the *RunTime* in the number of variables that actually exist. Instead, in the case of the variables that are been sent, those remain in the graphic because are actually created in the Draft document (128 variables created).

This is the main reason why it was not possible to perform the simulations with a single RunTime document, so a document for each number of variables was created.

Through the python program it was possible also to automate the creation of the *.sib* file by analysing the script structure of a *RunTime* document used as a model. In this way, for instance, if the number of variables of the *.sib* document was 15 the plot contains:

- 15 D signals.

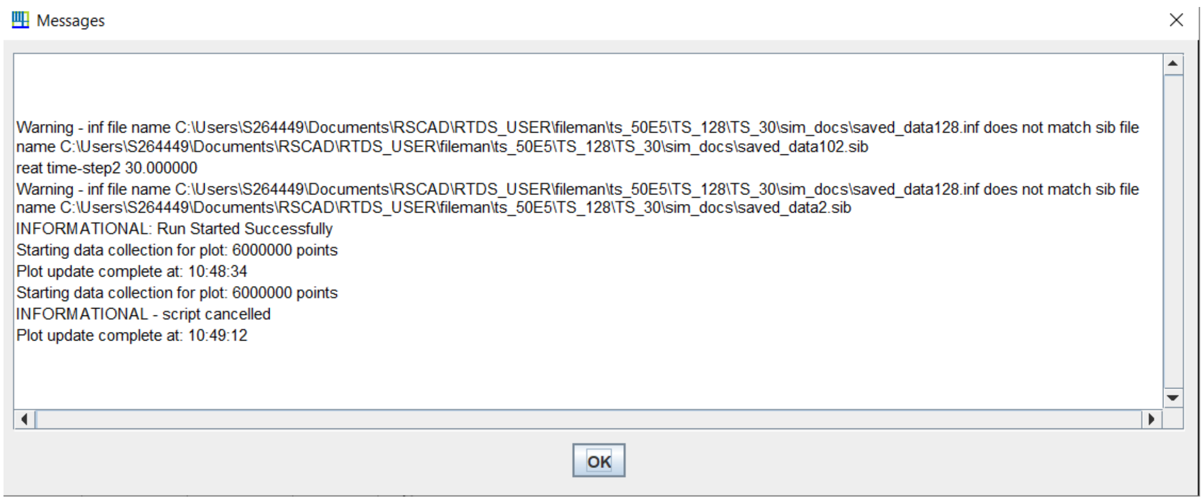
- 15 signals from_aurora1_i.
- 15 signals from_aurora2_i.

To perform several simulations automatically, the RunTime possess a feature that allows to develop a script and then through the RunTime this Script is opened and run. The deployed script is the following:

```
float result;
string step;
result = GetTimeStep();
fprintf (stdmsg,"reat time-step2 %f", result);

for(int a=2;a<129;a++){
        SetSlider "DraftVariables : numVar" = a;
        LoadBatch "saved_data"::itoa(a)::".sib";
        SetSwitch "Subsystem #1 : CTLs : Inputs : SW1" = 1;
        Start;
        UpdatePlots;
        SetSwitch "Subsystem #1 : CTLs : Inputs : SW1" = 0;
        Stop;
        step=ftoa(result);
        SavePlotToCSV "Subsystem
#1","C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\TS_128\TS_"::itoa(step)::"\
CV_docs\var"::itoa(a)::"ts"::itoa(step)::".csv";
        fprintf (stdmsg,"successfull num var: %f", a);
}
```

The main algorithms used to develop the script were: GetTimeStep, LoadBatch, SetSlider, SetSwitch and SavePlotToCSV. The GetTimeStep allowed to verify through the RunTime Message Area the actual timestep been simulated. The RunTime Message Area is a very useful tool to monitor the simulations. Figure 20 shows how the Message Area shows the information.



```
Warning - inf file name C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\TS_128\TS_30\sim_docs\saved_data128.inf does not match sib file
name C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\TS_128\TS_30\sim_docs\saved_data102.sib
reat time-step2 30.000000
Warning - inf file name C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\TS_128\TS_30\sim_docs\saved_data128.inf does not match sib file
name C:\Users\S264449\Documents\RSCAD\RTDS_USER\fileman\ts_50E5\TS_128\TS_30\sim_docs\saved_data2.sib
INFORMATIONAL: Run Started Successfully
Starting data collection for plot: 6000000 points
Plot update complete at: 10:48:34
Starting data collection for plot: 6000000 points
INFORMATIONAL - script cancelled
Plot update complete at: 10:49:12
```

Figure 20. RunTime Message Area.

The *LoadBatch* algorithm permitted to import a new *RunTime* document to perform the following steps of the simulation. In this way, it was possible to simulate and save the data collection of different number of variables each time. The SetSlider is the command that sets the number of variables to be passed to the *rts_draft_var* in the draft document. The

SetSwitch oversaw setting the switch first at 1 and then change it to 0 to start the collection of data. Finally, the *SavePlotToCSV* is the command that saves the collected data in the specified path.

For each simulation, the system has a maximum memory of 64 MB, which implies a limited number of points to be collected on a plot. Initially, the simulation times were calculated based on the performed steps on the simulation. The performed steps on the simulation is at the same time based on the maximum time-step reached when simulating for the biggest number of variables and the smallest time step (128 variables and 7 µs), using equation (10).

$$Simulation\ steps = \frac{Max_{simTime}\ x\ n_p}{T_s} \tag{10}$$

$$Plotted\ points = Simulation\ steps\ x\ 3 \tag{11}$$

Where n_p stands for the number of variables the simulation is been performed for and T_s stands for the step time. $Max_{simTime}$ is the maximum simulation time reached with the conditions stated before. The plotted points are computed multiplying the simulation time by three as stated in Equation (11).

The system failed through the script when simulating more than 6 100 000 points. This happened when the amount of data to be collected was bigger than the memory available. The raised error was of data overflow that consequently caused the breaking of the script process. Several simulations iterating the number of plotted points where deployed, starting from 6 900 000 plotted points and decreasing the number of plotted points continuously. On each of these iterations the *.sib* documents had to be modify and created again across the python program using the new iteration variables.

In order to reach simulation results with statistic congruency (simulating approximately the same number of steps) it was imposed a number of points of 6 000 000, which is translated into 2 000 000 steps. In the section, SIMULATION TIME COMPUTATION table shows a table with all the simulation times corresponding to the imposed plotted points for each simulation case.

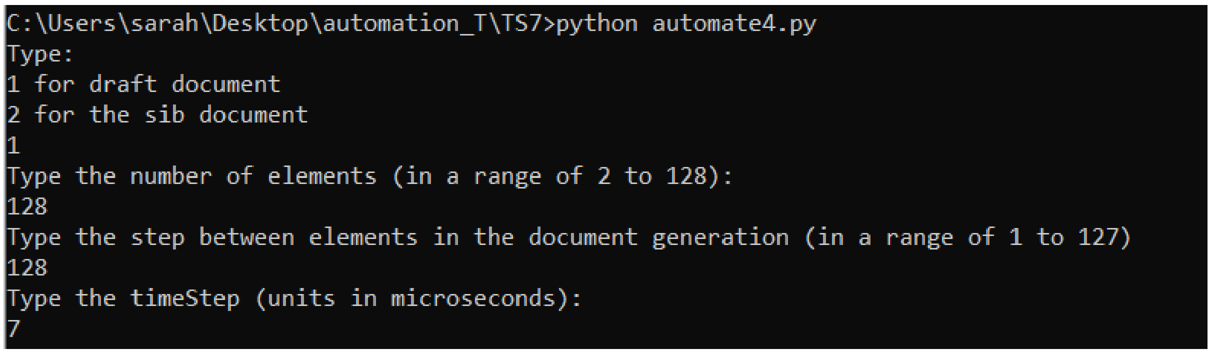
Therefore, another reason why it was necessary to create a *.sib* document for each simulation was to assure the congruency across the CSV document generation steps of the different timestep simulations. The simulation time can only be changed through the *.sib* document. The python program (APENDIX AAPENDIX A) computes the simulation time based on the timestep entered by the user when running the program, and creates the .sib document with the corresponding simulation time.

As stated in Latency Calculation sub-section, in the Methodology chapter, the created model is based on a python program that generates the necessary documents to deploy the simulation test case. Ten step times where tested as shown in the computation table in SIMULATION TIME COMPUTATION, in the APENDIX B. For each of these ten step times, it was necessary to create a *.dft* document and 128 .sib documents.

When running the python program, four queries will arise as shown in Figure 21. The first one asks the type of document to be generated (draft or sib document). The second question concerns the quantity of elements (timers) to be considered in the document creation.

The third query asks the step between the creation of documents. For instance, if it is intended to create only one *.dft* document of 128 timers, the answer in this query should be 128. This feature allows to spare time when only one document is planned to be created. Since the Aurora block needs at least 2 points to communicate, the range was fixed from 2 to 128 timers.

Finally, the program asks the timestep (in microseconds) to set it in the document generation. From the experiments, the allowable timesteps is up to 7 µs (stating this timestep as the smallest). The program will take the entered data by the user and will generate the necessary files with the stated configuration on the model specifications.


```
C:\Users\sarah\Desktop\automation_T\TS7>python automate4.py
Type:
1 for draft document
2 for the sib document
1
Type the number of elements (in a range of 2 to 128):
128
Type the step between elements in the document generation (in a range of 1 to 127)
128
Type the timeStep (units in microseconds):
7
```
Figure 21. Generation of documents by the python program

To perform the simulation on the RTDS server, it was created a folder for each timestep. Each of these folders contained another two folders: CV_docs and sim_docs (Figure 22). The *sim_docs* contained the *.dft* document, where it was compiled in order to have all the documents generated after the compiling process. The *.sib* documents, after been generated by the python program, must be place in this folder, as well as the script file.

The *CV_docs* folder was created to save the collected data when the script was running. The script is set to search the correct path and save the *.csv* documents (this path can be modified depending on the user's needs).

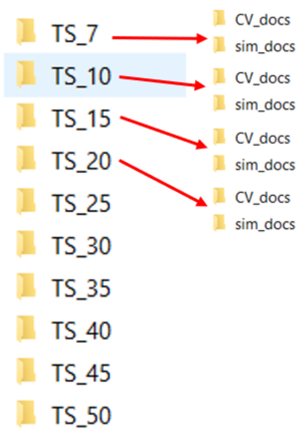


Figure 22. Simulation folder distribution.

Finally, from the RSCAD file manager, one of the RunTime documents in the timestep folder intended to be simulated, needs to be opened. In the RunTime, in order to start the script, it is necessary to open the script504.scr. After selecting the script, the play button must be pressed, and the simulations from 2 to 128 variables are performed and the data is saved automatically. Every time a simulation for a different timestep is to be deployed, the previous procedure must be repeated.

When going through the *.sib* documents, a warning message appears stating that the name of the compiled document does not correspond to the RunTime document. This warning does not affect the simulation performance.

The results obtained were a variation of the transmitted message time-delay (from one port to another) of two timesteps, regardless of both the simulated variables quantity and the imposed timestep. Figure 23 shows a histogram where are plotted the delay-time (µs), the number of variables and the timestep (µs).

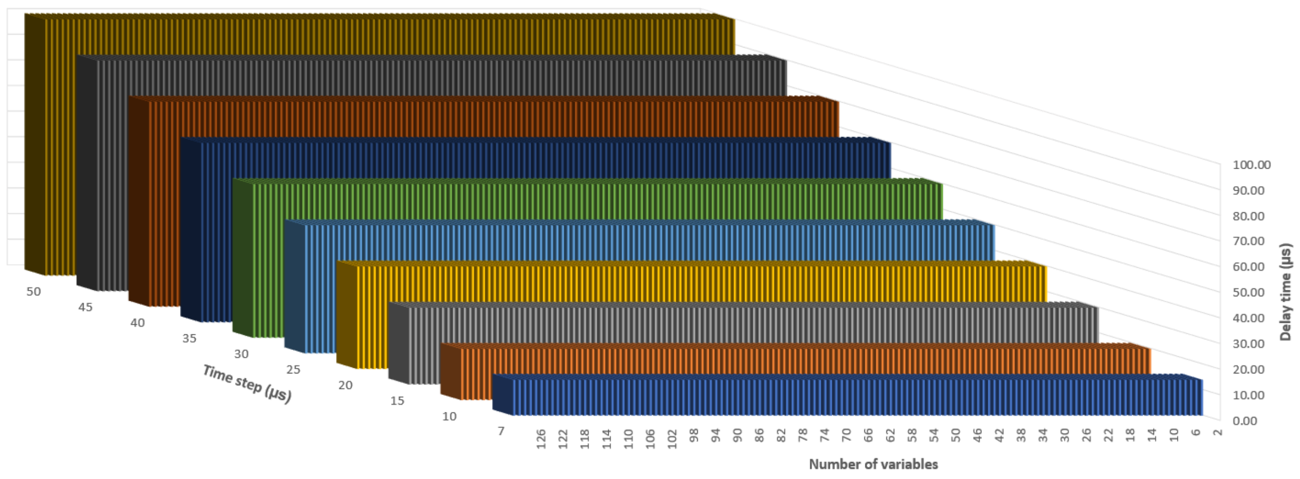


Figure 23. Time delay histogram.

Since the number of variables does not affect the behaviour of the time delay, the correspondent axis was deleted from the graphic, and converted into a two-axis graphic. From Figure 24, it is straight forward to determine that the behaviour of the time delay is completely linear.

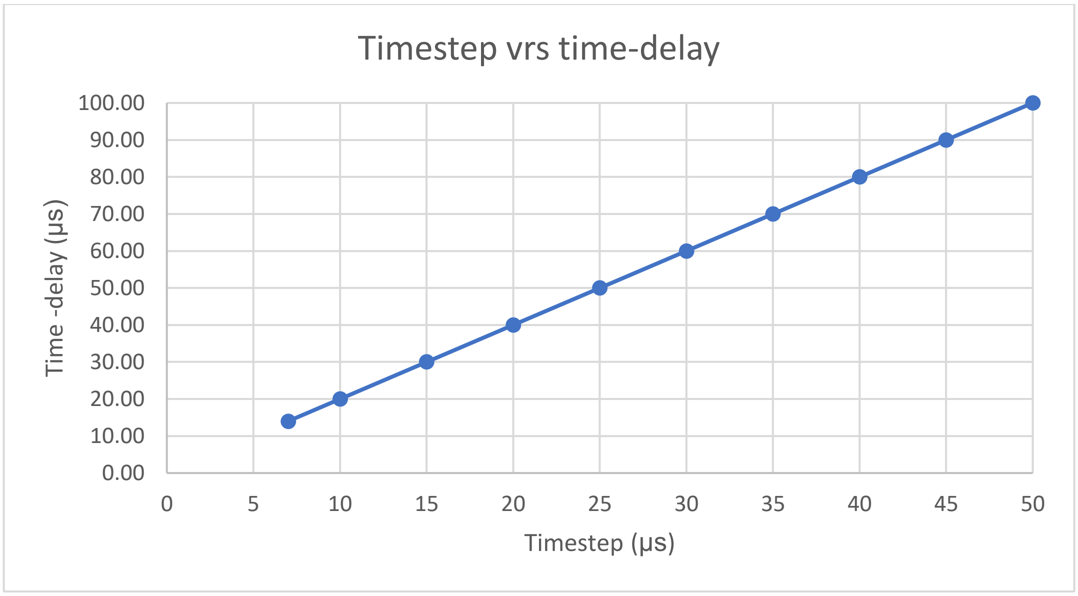


Figure 24. Two-axis plotting of the timestep.

5.2 Simple Case Study

5.2.1 ITM Stability and Accuracy Application to DRTS Co-simulation

The theoretical analysis was performed using a simple circuit with two resistances, considering the introduced delay by the communication from one port to the another.

The model was composed by two circuits. The first part of the circuit (Circuit A) was composed by a voltage generator (E) a resistance (Z_A) and a current generator which was in charge of providing the feedback to Circuit A. Circuit B is composed by a voltage generator remotely controlled by the Voltage A (V_A), and a resistance (Z_B). The measured current i_B is the feedback in Circuit A. Figure 25 shows the general previously described model of the system.

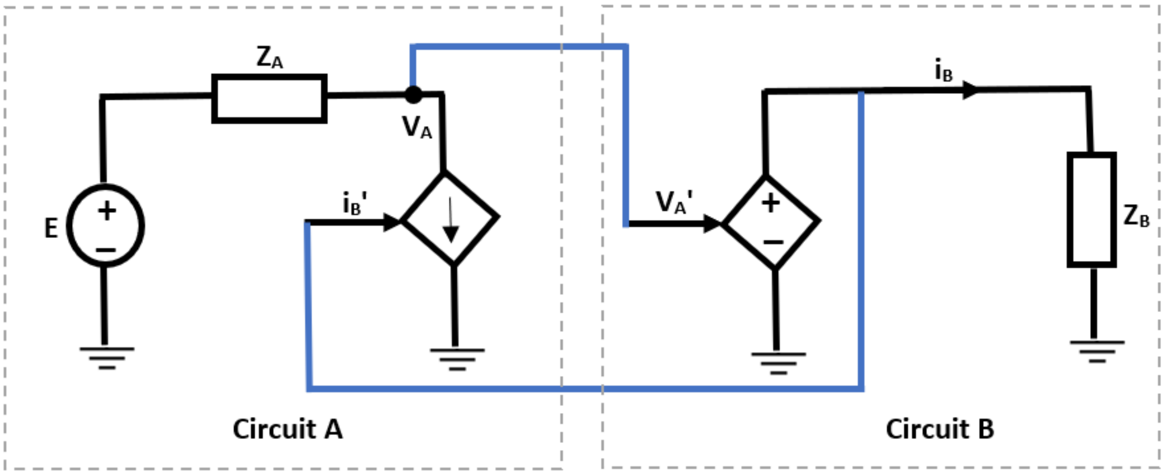


Figure 25. General model of the delayed system implemented.

The blue line shown in Figure 25 illustrates the delay on the data transmission. In the test case (Latency Calculation section), it was proved that the transmission time is two times the step time.

Figure 26 shows the equivalent block diagram of the system.

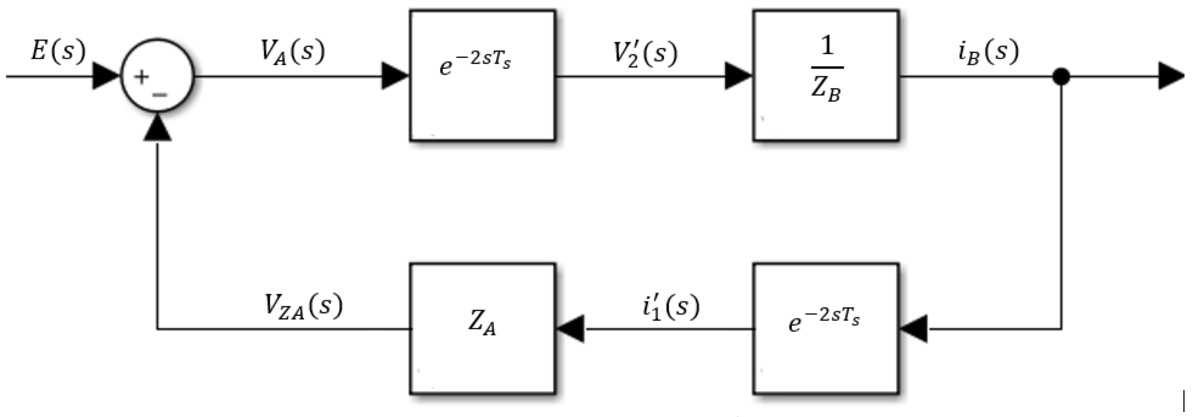


Figure 26. Equivalent block diagram of the system

In Figure 26, T_s is the imposed timestep of the real time simulator. In order to analyse the system stability, it is necessary to analyse the open loop transfer function. The open loop transfer function can be reached using the block diagram shown in Figure 26, but the feedback connection must be neglected. Equation (12) presents the open loop transfer function.

$$G_{OL} = \frac{Z_A}{Z_B} e^{-4sT_s}$$

(12)

To analyse the system response, a variable k was introduced, where $k = \frac{Z_A}{Z_B}$, which makes Equation (13) to become:

$$G_{OL} = ke^{-4sT_s}$$

(13)

The complete transfer function of the system is reached by Equations (14), (15), (16) and (17) as follows:

$$V_A = E - V_{ZA}$$

(14)

$$V_2' = V_A\, e^{-2sT_s}$$

(15)

$$i_B = \frac{V_2'}{Z_B} \rightarrow i_B = \frac{V_A\, e^{-2sT_s}}{Z_B}$$

(16)

$$i_1' = i_B\, e^{-sT_s}$$

(17)

Using the superposition principle, Equation (18) was reached:

$$V_{ZA} = Z_A * (i_1' + i_E),\ where\, i_E = 0$$

(18)

Using (14), (16), (17) and (18), Equation (19) is obtained as follows:

$$E - \frac{V_A * e^{-2sT_S}\, Z_A\, e^{-2sT_S}}{Z_B} = V_A \;\rightarrow\; V_A = \frac{Z_B\, E}{Z_B + Z_A * e^{-2sT_S}} \tag{19}$$

Finally, the complete transfer function is obtained in Equation (20):

$$\boldsymbol{G_{ALL}} = \frac{\mathbf{1}}{\mathbf{1} + \frac{\boldsymbol{Z_A}}{\boldsymbol{Z_B}} * \boldsymbol{e^{-4sT_s}}} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{G_{OL}}} \tag{20}$$

To analyse the system stability, the Nyquist diagram was used to determine for which values of *k* the system would be stable. Figure 27 and Figure 28 show the Nyquist diagrams when using a value of *k* bigger and smaller than 1 respectively.

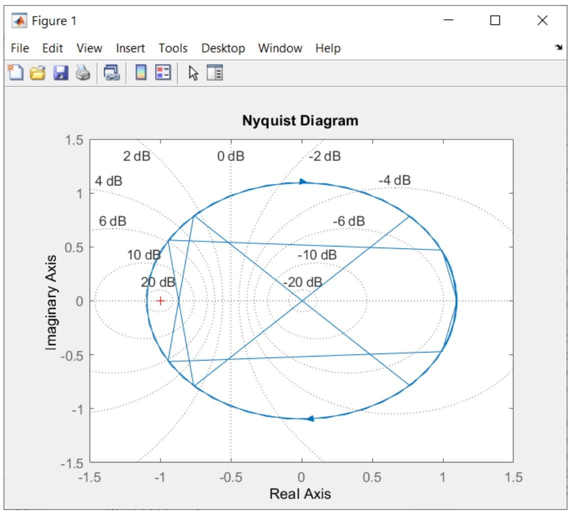


Figure 27. Nyquist diagram when k=1.1

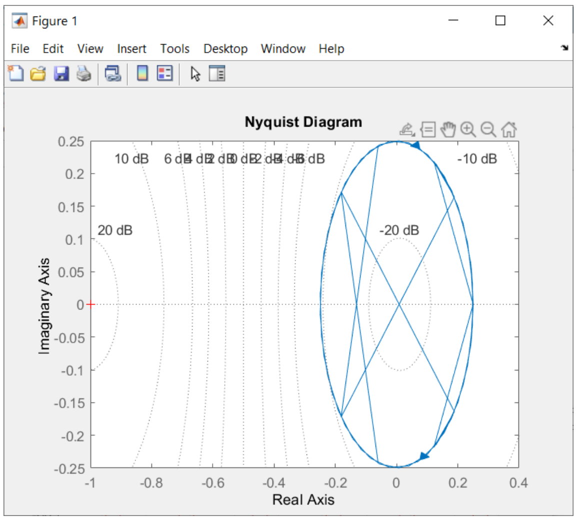


Figure 28. Nyquist diagram when k=0.25

The value of k reverberates directly on the size of the circle. When k is smaller than one, the circle tends to get narrower, instead when k tends to go further one, the circle widens. When k gets closer to 1, the circle gets closer to the transfer function zero solution, and when k is equal or bigger than one, the circle contains the zero, so the system becomes unstable.

In each case, it was also analysed the system response and the theoretical time to reach the stability.

Figure 29 and Figure 30 show the different system responses when k=0.98 and when k=0.83 respectively, with a timestep in both cases of 50 µs.

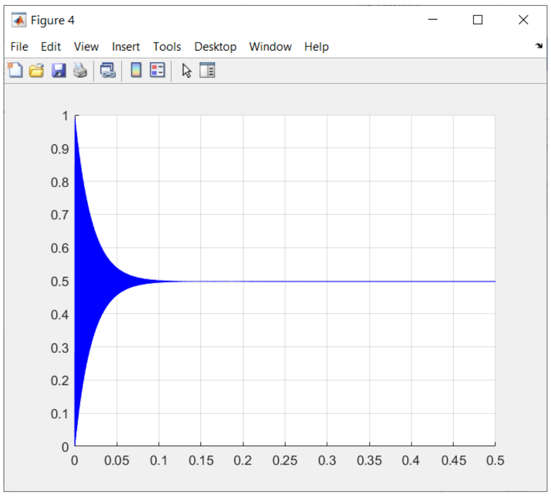


Figure 29. System response when k=0.98.

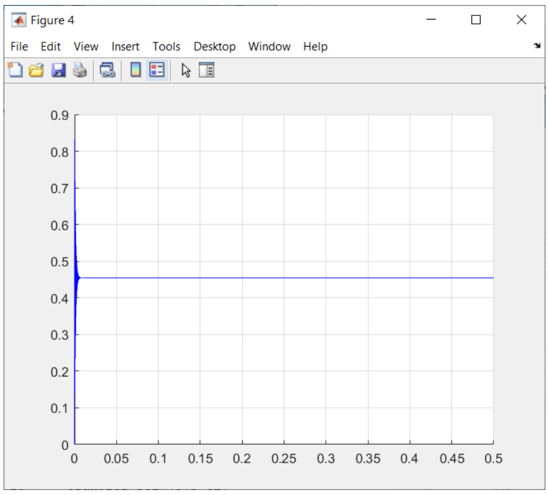


Figure 30. System response when k=0.83.

Therefore, different cases were analysed, by varying the value of k and also varying the time step to determine the correlation and the affectations that these variables could have on the system. Table 4 shows the theoretical system response analysed graphically by the plotting performed through MATLAB when iterating the timestep and the different values of k.

The ideal transient behaviour of all systems can be modelled with a second-degree transfer function when the system reference signal is a step function. To analyse the V_A response with respect to the different timesteps and k values, the maximum overshoot and the settling time ($t_{s,5\%}$) are the main parameters to judge the signal stabilization behaviour. The ideal behaviour is when the overshoot is small and the settling time is fast. The output signal is considered to have reached the settling time when the signal oscillations doesn't surpass the upper and bottom thresholds of 5% the tending to infinity output signal (y_∞).

Table 4. Theoretical system response transient analysis

T_{STEP}	Z_A	Z_B	K	Y_{MAX}	Y_∞	OVERSHOOT	$T_{S,5\%}$	V_{OUT}
(µS)	(Ω)	(Ω)	(Z_A/ Z_B)	(V)	(V)		(s)	(V)
500	50	500	0.1000	0.1	0.0909	0.1000	0.0040	93.02
500	50	50.5	0.9901	0.9804	0.4975	0.9706	0.6020	65.97
150	50	500	0.1000	0.1	0.0909	0.1000	0.0012	91.10
150	50	200	0.2500	0.25	0.2000	0.2500	0.0018	80.33
150	50	60	0.8333	0.8333	0.4546	0.8332	0.0102	54.90
150	50	51	0.9804	0.9804	0.4950	0.9804	0.0912	51.00
150	50	50.5	0.9901	1	0.4955	1.0182	0.1878	51.77
100	50	500	0.1000	0.1	0.0909	0.1000	0.0008	90.99
100	50	200	0.2500	0.25	0.2000	0.2500	0.0012	80.15
100	50	60	0.8333	0.8333	0.4546	0.8332	0.0068	54.70
100	50	51	0.9804	0.9804	0.4951	0.9804	0.0612	50.62
100	50	50.5	0.9901	1	0.4951	1.0200	0.1212	50.50
50	50	500	0.1000	0.1	0.0909	0.1000	0.0004	90.93
50	50	200	0.2500	0.25	0.2000	0.2500	0.0006	80.04
50	50	60	0.8333	0.8333	0.4545	0.8334	0.0034	54.58
50	50	51	0.9804	0.9804	0.4951	0.9804	0.0306	50.53
50	50	50.5	0.9901	1	0.4975	1.0100	0.0604	50.31
5	50	500	0.1000	0.1	0.0909	0.1000	4.00E-05	90.91
5	50	200	0.2500	0.25	0.2000	0.2500	4.00E-05	80.00
5	50	60	0.8333	0.8333	0.4546	0.8332	3.20E-04	54.55
5	50	51	0.9804	0.9804	0.4951	0.9804	3.00E-03	50.50
5	50	50.5	0.9901	0.9901	0.4975	0.9902	6.01E-02	50.25
50	**60**	**50**	**1.2**					

Looking at the obtained values in Table 4, if keeping constant the timestep, but varying k, when k is closer to one, both the overshoot and the settling time are bigger. In a diverse manner, if the timestep varies, but the k value is kept constant, when the timestep is smaller, the settling time also is smaller. In this last case, the overshoot barely changes. From these theoretical results, it is evident that the critical cases to analyse are when k is very close to one (Z_B=50.5) and when the timestep is very big (T_{step} = 500 µs). Opposite to the previous statement, the least critical k value is when k is smaller (Z_B=500).

5.2.2 Power Electric Result

To perform experimental analysis, three stages where set to determine the differences between the models. The naturally coupled system, the decoupled system but without using the Aurora link, and a third one also decoupled but performing the communication trough the Aurora link.

For the first stage, the naturally coupled circuit (Circuit 1) was created as shown in Figure 31. The usage of nodes was implemented to graph the voltage input, the voltage in A (Va) and the current by the usage of the current ammeter.

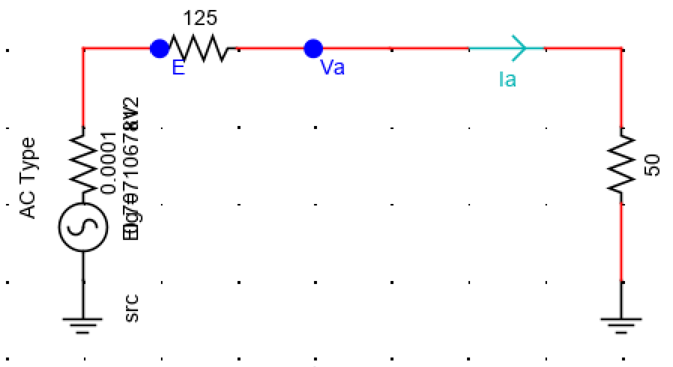
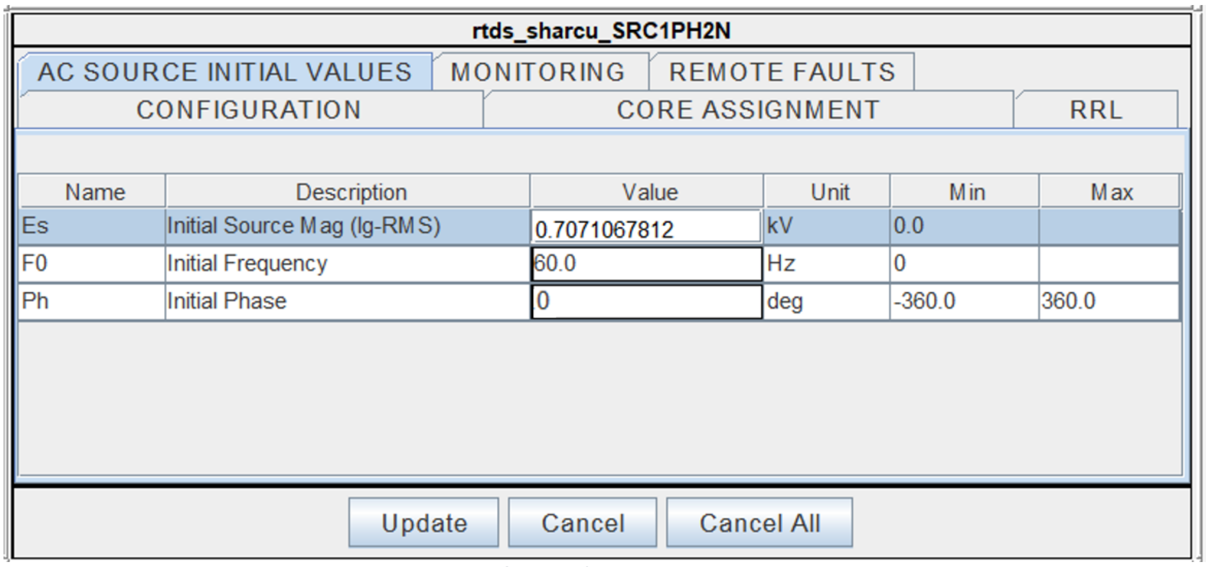


Figure 31. Circuit 1 model.

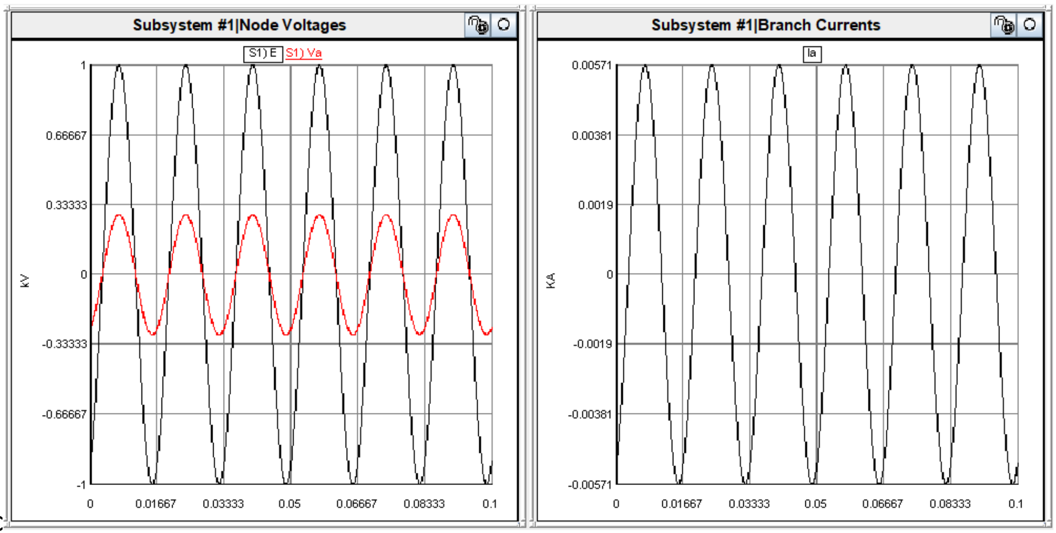
All the components were selected from the *Power System* library, in the subsection of single phase components. Since the voltage generator algorithm constrained to specified the RMS voltage, the imposed input voltage was 0.7071 V_{RMS}, to obtain a 1 V_{pp} in E as shown in Figure 32. The initial phase was selected as 60 Hz and the initial phase 0 deg. Regarding the voltage generator resistance, since it was included already in the algorithm, a value of 0.0001 Ω was imposed because it is suficiently small to consider its influence negligible on the measured voltage on E.



Name	Description	Value	Unit	Min	Max
Es	Initial Source Mag (lg-RMS)	0.7071067812	kV	0.0	
F0	Initial Frequency	60.0	Hz	0	
Ph	Initial Phase	0	deg	-360.0	360.0

Figure 32. Setup of the voltage generator.

The results of the circuit illustrated in Figure 31, using the parameters: Z_A=125 Ω and Z_B=50 Ω are shown in Figure 33 simulating with a timestep of 50 µs. This figure shows the graphic of the input voltage (E), Va and the current of the circuit (Ia).



C

Figure 33. Result of circuit 2 when when Z_A=125 Ω and Z_B=50 Ω.

Figure 34 shows the graphic of the results when Z_A=50 Ω and Z_B=500 Ω.

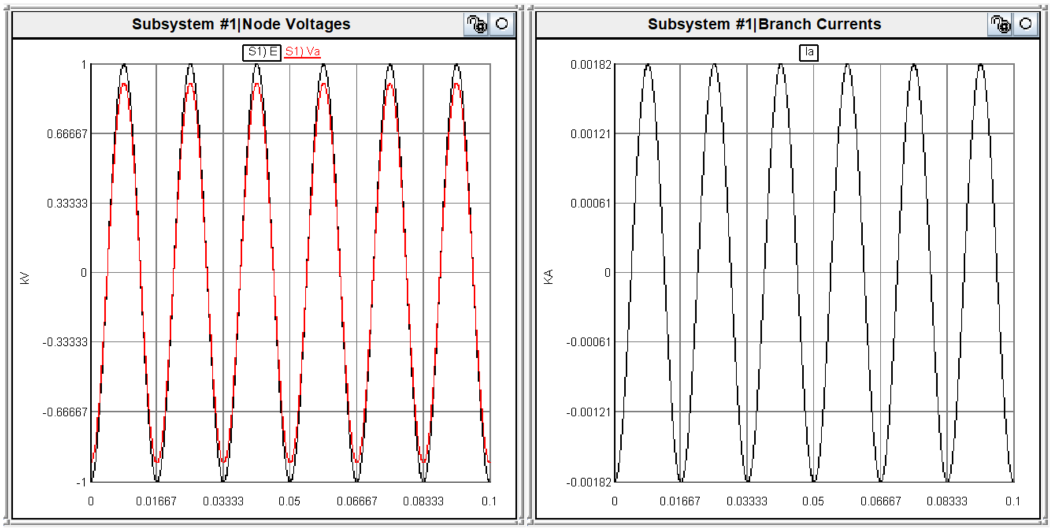


Figure 34. Result of circuit 2 when Z_A=50 Ω and Z_B=500 Ω.

The second stage consisted in dividing the circuit in two parts. For this purpose, a new configuration system (Circuit 2) was created as shown in Figure 35. The first part of the circuit (Circuit A) where Va was exported to the second circuit (Circuit B). The measured value of Va is introduced in the remote-controlled voltage generator in Circuit B. The measured value of current (Ia) measured in Circuit B is also exported and introduced in Circuit A.

The problem in this model is that the simulator doesn't allow to directly connect an exported signal to the remote-controlled current injector algorithm. The current injection/source algorithm requires the input current to be a control signal. This is the reason why a gain of 1 was used between Ia and Iin, to use Iin signal as input in the current injection/source in Circuit A.

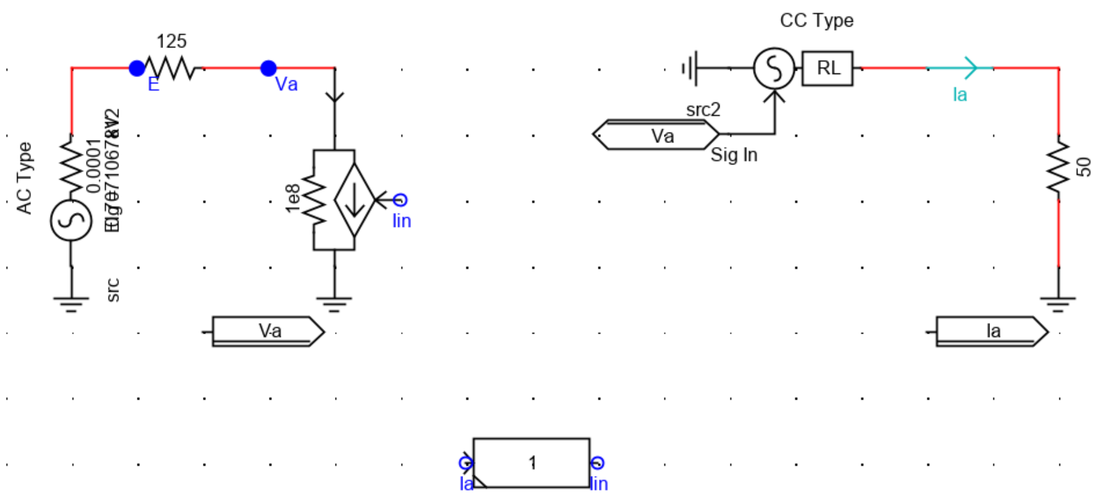


Figure 35. Circuit 2 model.

Even though in the case of Circuit 2 model the communication protocol is not being implemented, when trying to simulate the circuit with the parameters shown in Figure 35, Z_A=125 Ω and Z_B=50 Ω (which makes the system unstable), the RTDS doesn´t allow to deploy the simulation. This means that there is a delay in this system that makes it unstable like in the third circuit, but in this case the delay is neither measurable nor determinable due to the characteristics of the circuit on the simulator.

Instead, when Z_A=50 Ω and Z_B=500 Ω, parameters that make the system stable, it possible to appreciate the RunTime graphed results shown in Figure 36 with a timestep of 50 µs. Figure 37 shows the results when Z_A=50 Ω and Z_B=51 Ω.

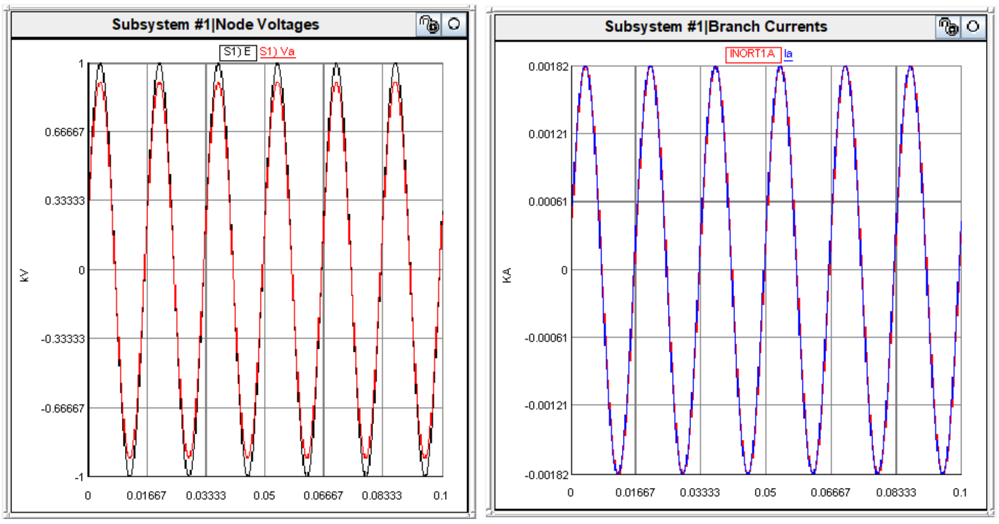


Figure 36. Result of circuit 3 when Z_A=50 Ω and Z_B=500 Ω.

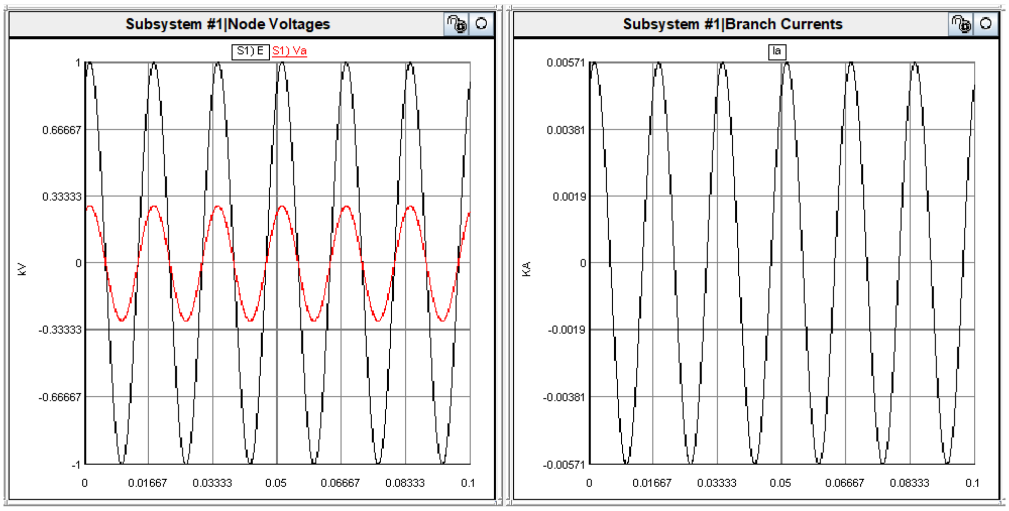


Figure 37. Result of circuit 3 when Z_A=50 Ω and Z_B=51 Ω.

The third stage was deployed dividing the electrical circuit in two parts. In this case the communication is performed using Aurora protocol. Two variables are sent/receive through the protocol. The first variable is the voltage (V_A) that is an output of circuit A, but an input for Circuit B. The second variable is Ia that is an output of Circuit B and is sent through the protocol to feed the current injection source in Circuit A. For this purpose, it was also necessary to create a new configuration as shown in Figure 38.

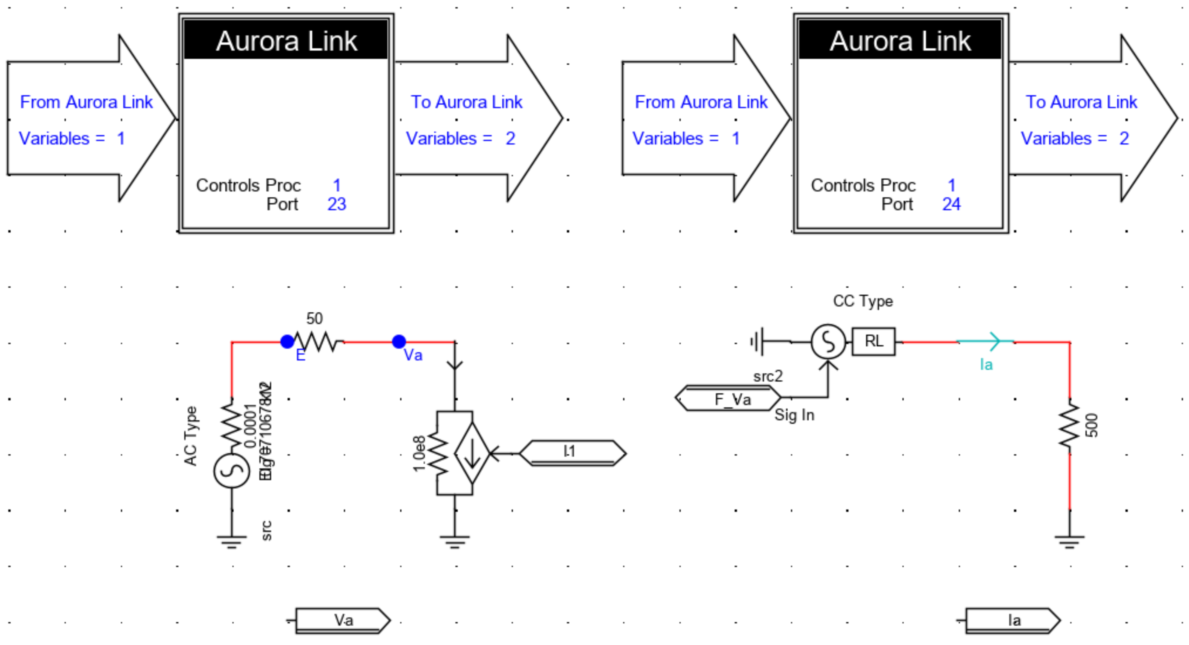
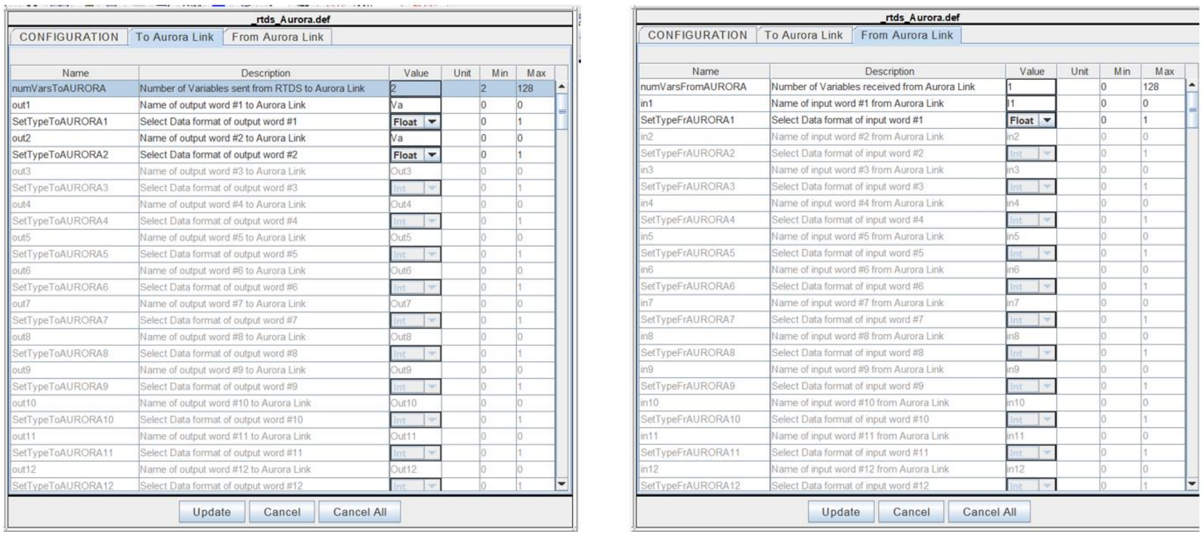


Figure 38. Decoupled system performing communication between the subsystem by the Aurora link.

The configuration of the first Aurora Link, that corresponds to the data sent from Port 23 is shown in Figure 39. The parameter Va is sent twice because the minimum amount of data allowed to send is two variables.



_rtds_Aurora.def

CONFIGURATION | To Aurora Link | From Aurora Link

Name	Description	Value	Unit	Min	Max
numVarsToAURORA	Number of Variables sent from RTDS to Aurora Link	2		2	128
out1	Name of output word #1 to Aurora Link	Va		0	0
SetTypeToAURORA1	Select Data format of output word #1	Float		0	1
out2	Name of output word #2 to Aurora Link	Va		0	0
SetTypeToAURORA2	Select Data format of output word #2	Float		0	1
out3	Name of output word #3 to Aurora Link	Out3		0	0
SetTypeToAURORA3	Select Data format of output word #3	Int		0	1
out4	Name of output word #4 to Aurora Link	Out4		0	0
SetTypeToAURORA4	Select Data format of output word #4	Int		0	1
out5	Name of output word #5 to Aurora Link	Out5		0	0
SetTypeToAURORA5	Select Data format of output word #5	Int		0	1
out6	Name of output word #6 to Aurora Link	Out6		0	0
SetTypeToAURORA6	Select Data format of output word #6	Int		0	1
out7	Name of output word #7 to Aurora Link	Out7		0	0
SetTypeToAURORA7	Select Data format of output word #7	Int		0	1
out8	Name of output word #8 to Aurora Link	Out8		0	0
SetTypeToAURORA8	Select Data format of output word #8	Int		0	1
out9	Name of output word #9 to Aurora Link	Out9		0	0
SetTypeToAURORA9	Select Data format of output word #9	Int		0	1
out10	Name of output word #10 to Aurora Link	Out10		0	0
SetTypeToAURORA10	Select Data format of output word #10	Int		0	1
out11	Name of output word #11 to Aurora Link	Out11		0	0
SetTypeToAURORA11	Select Data format of output word #11	Int		0	1
out12	Name of output word #12 to Aurora Link	Out12		0	0
SetTypeToAURORA12	Select Data format of output word #12	Int		0	1

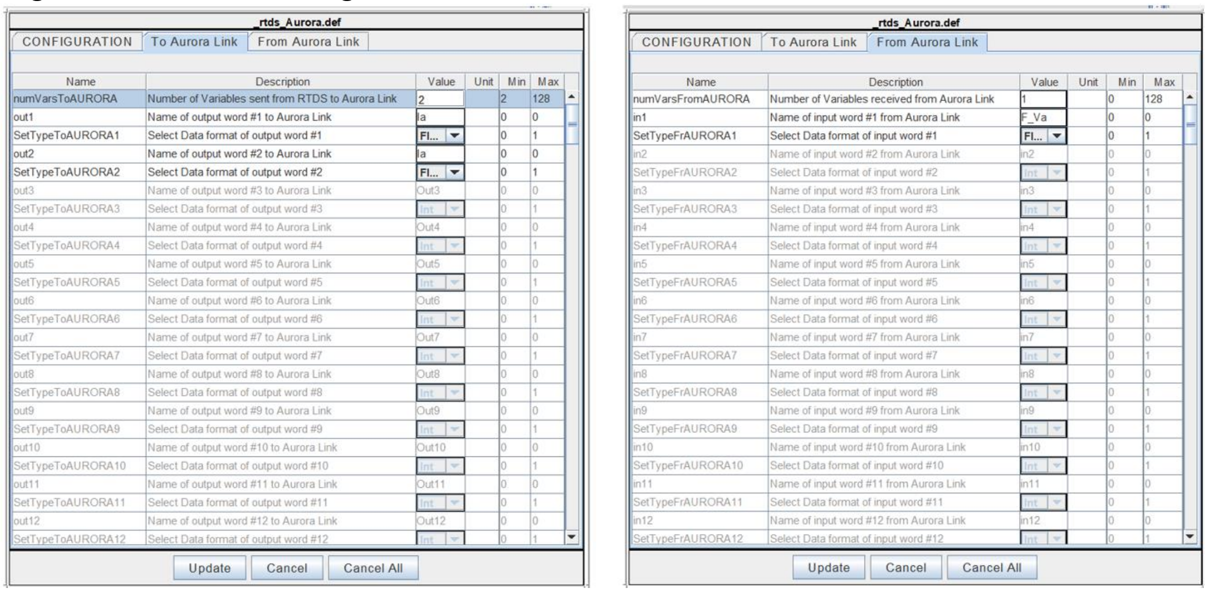
_rtds_Aurora.def

CONFIGURATION | To Aurora Link | From Aurora Link

Name	Description	Value	Unit	Min	Max
numVarsFromAURORA	Number of Variables received from Aurora Link	1		0	128
in1	Name of input word #1 from Aurora Link	I1		0	0
SetTypeFrAURORA1	Select Data format of input word #1	Float		0	1
in2	Name of input word #2 from Aurora Link	in2		0	0
SetTypeFrAURORA2	Select Data format of input word #2	Int		0	1
in3	Name of input word #3 from Aurora Link	in3		0	0
SetTypeFrAURORA3	Select Data format of input word #3	Int		0	1
in4	Name of input word #4 from Aurora Link	in4		0	0
SetTypeFrAURORA4	Select Data format of input word #4	Int		0	1
in5	Name of input word #5 from Aurora Link	in5		0	0
SetTypeFrAURORA5	Select Data format of input word #5	Int		0	1
in6	Name of input word #6 from Aurora Link	in6		0	0
SetTypeFrAURORA6	Select Data format of input word #6	Int		0	1
in7	Name of input word #7 from Aurora Link	in7		0	0
SetTypeFrAURORA7	Select Data format of input word #7	Int		0	1
in8	Name of input word #8 from Aurora Link	in8		0	0
SetTypeFrAURORA8	Select Data format of input word #8	Int		0	1
in9	Name of input word #9 from Aurora Link	in9		0	0
SetTypeFrAURORA9	Select Data format of input word #9	Int		0	1
in10	Name of input word #10 from Aurora Link	in10		0	0
SetTypeFrAURORA10	Select Data format of input word #10	Int		0	1
in11	Name of input word #11 from Aurora Link	in11		0	0
SetTypeFrAURORA11	Select Data format of input word #11	Int		0	1
in12	Name of input word #12 from Aurora Link	in12		0	0
SetTypeFrAURORA12	Select Data format of input word #12	Int		0	1

Figure 39. Configuration of the Aurora Link, Port 23.

Figure 40 shows the configuration of the second Aurora block.



_rtds_Aurora.def

CONFIGURATION | To Aurora Link | From Aurora Link

Name	Description	Value	Unit	Min	Max
numVarsToAURORA	Number of Variables sent from RTDS to Aurora Link	2		2	128
out1	Name of output word #1 to Aurora Link	Ia		0	0
SetTypeToAURORA1	Select Data format of output word #1	Fl...		0	1
out2	Name of output word #2 to Aurora Link	Ia		0	0
SetTypeToAURORA2	Select Data format of output word #2	Fl...		0	1
out3	Name of output word #3 to Aurora Link	Out3		0	0
SetTypeToAURORA3	Select Data format of output word #3	Int		0	1
out4	Name of output word #4 to Aurora Link	Out4		0	0
SetTypeToAURORA4	Select Data format of output word #4	Int		0	1
out5	Name of output word #5 to Aurora Link	Out5		0	0
SetTypeToAURORA5	Select Data format of output word #5	Int		0	1
out6	Name of output word #6 to Aurora Link	Out6		0	0
SetTypeToAURORA6	Select Data format of output word #6	Int		0	1
out7	Name of output word #7 to Aurora Link	Out7		0	0
SetTypeToAURORA7	Select Data format of output word #7	Int		0	1
out8	Name of output word #8 to Aurora Link	Out8		0	0
SetTypeToAURORA8	Select Data format of output word #8	Int		0	1
out9	Name of output word #9 to Aurora Link	Out9		0	0
SetTypeToAURORA9	Select Data format of output word #9	Int		0	1
out10	Name of output word #10 to Aurora Link	Out10		0	0
SetTypeToAURORA10	Select Data format of output word #10	Int		0	1
out11	Name of output word #11 to Aurora Link	Out11		0	0
SetTypeToAURORA11	Select Data format of output word #11	Int		0	1
out12	Name of output word #12 to Aurora Link	Out12		0	0
SetTypeToAURORA12	Select Data format of output word #12	Int		0	1

_rtds_Aurora.def

CONFIGURATION | To Aurora Link | From Aurora Link

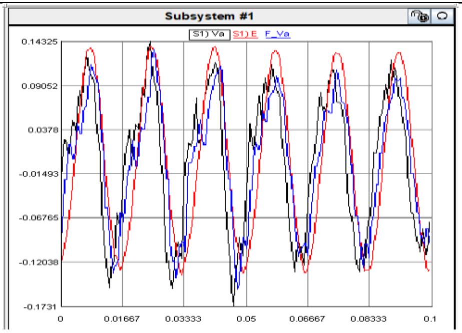
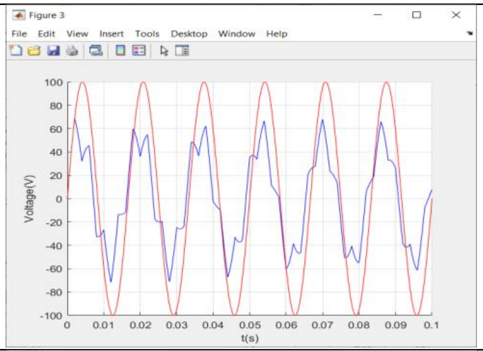
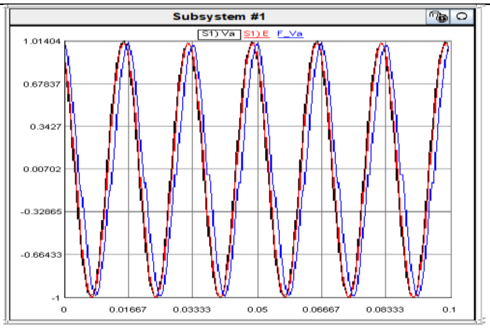
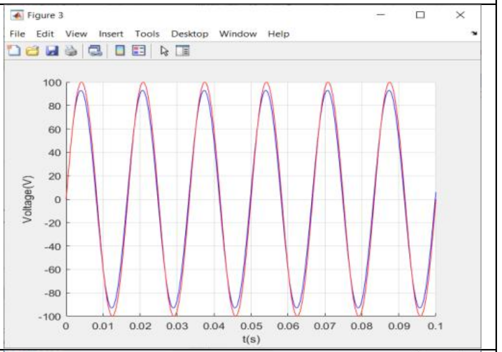
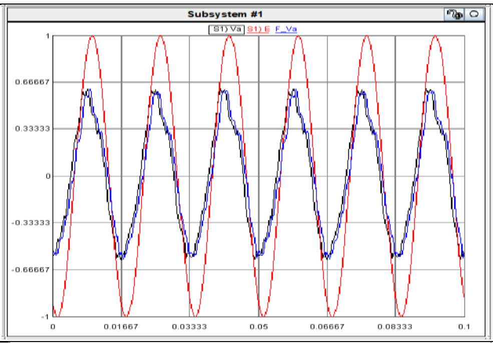
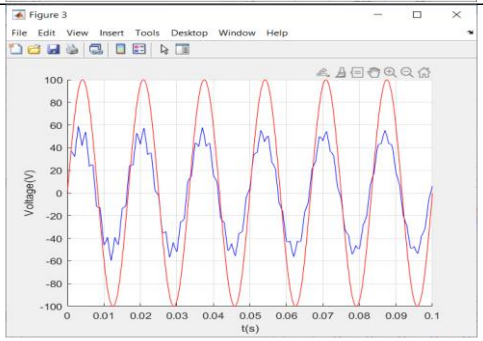
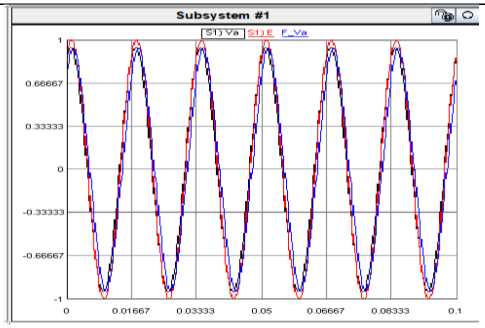
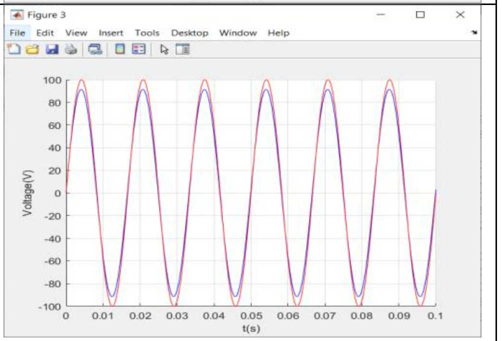
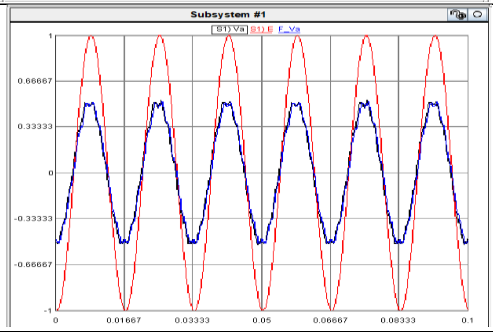
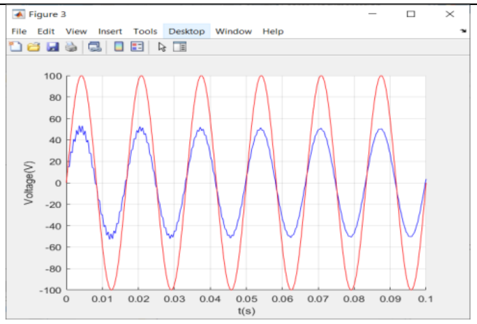
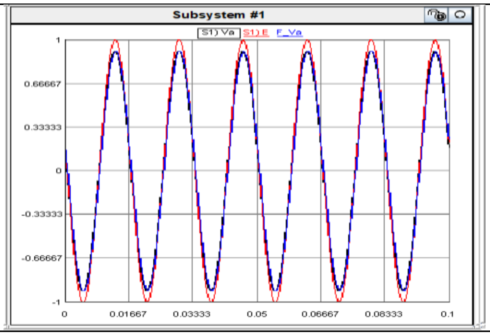
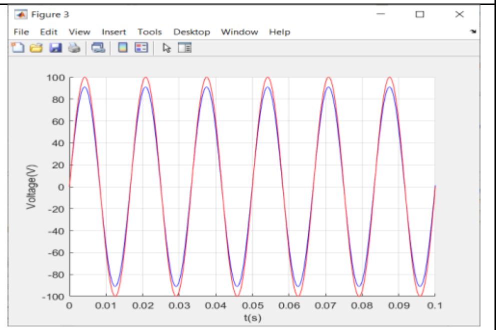
Name	Description	Value	Unit	Min	Max
numVarsFromAURORA	Number of Variables received from Aurora Link	1		0	128
in1	Name of input word #1 from Aurora Link	F_Va		0	0
SetTypeFrAURORA1	Select Data format of input word #1	Fl...		0	1
in2	Name of input word #2 from Aurora Link	in2		0	0
SetTypeFrAURORA2	Select Data format of input word #2	Int		0	1
in3	Name of input word #3 from Aurora Link	in3		0	0
SetTypeFrAURORA3	Select Data format of input word #3	Int		0	1
in4	Name of input word #4 from Aurora Link	in4		0	0
SetTypeFrAURORA4	Select Data format of input word #4	Int		0	1
in5	Name of input word #5 from Aurora Link	in5		0	0
SetTypeFrAURORA5	Select Data format of input word #5	Int		0	1
in6	Name of input word #6 from Aurora Link	in6		0	0
SetTypeFrAURORA6	Select Data format of input word #6	Int		0	1
in7	Name of input word #7 from Aurora Link	in7		0	0
SetTypeFrAURORA7	Select Data format of input word #7	Int		0	1
in8	Name of input word #8 from Aurora Link	in8		0	0
SetTypeFrAURORA8	Select Data format of input word #8	Int		0	1
in9	Name of input word #9 from Aurora Link	in9		0	0
SetTypeFrAURORA9	Select Data format of input word #9	Int		0	1
in10	Name of input word #10 from Aurora Link	in10		0	0
SetTypeFrAURORA10	Select Data format of input word #10	Int		0	1
in11	Name of input word #11 from Aurora Link	in11		0	0
SetTypeFrAURORA11	Select Data format of input word #11	Int		0	1
in12	Name of input word #12 from Aurora Link	in12		0	0
SetTypeFrAURORA12	Select Data format of input word #12	Int		0	1

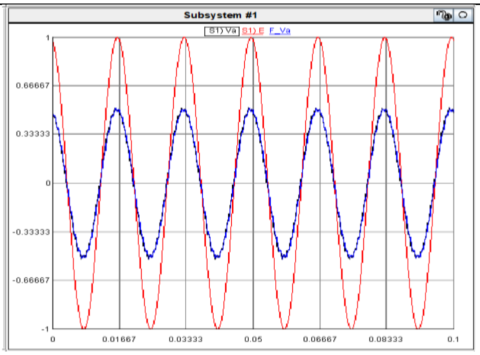
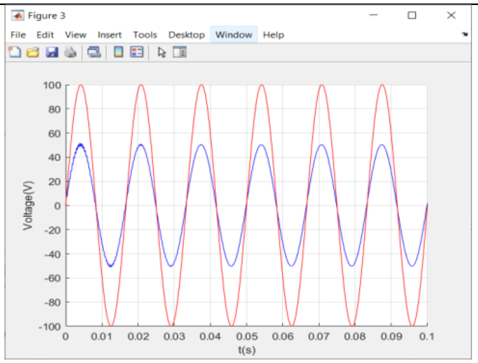
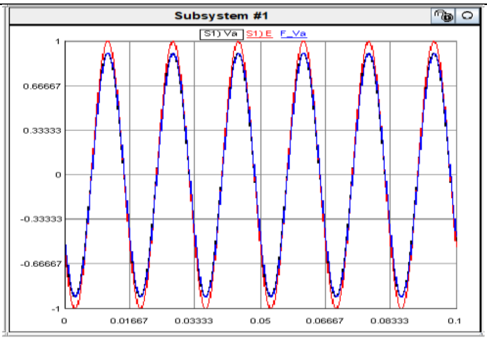
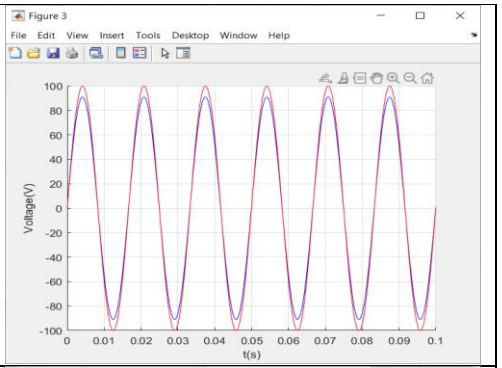
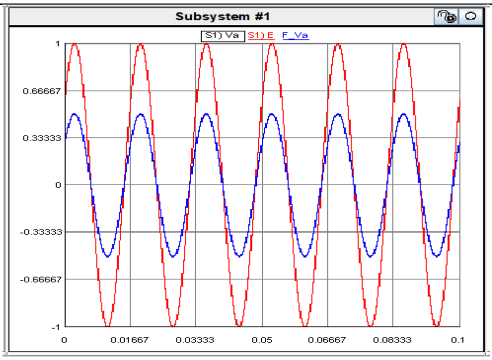
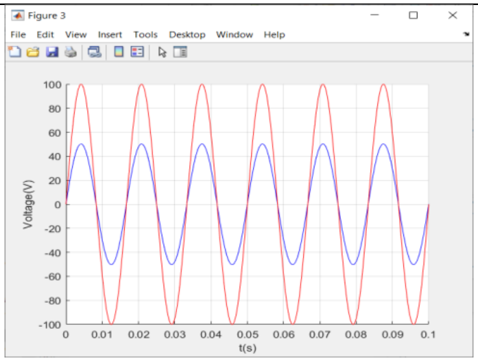
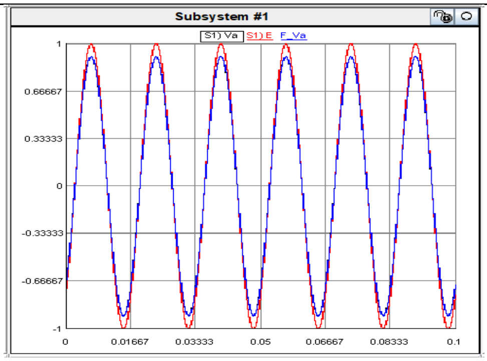
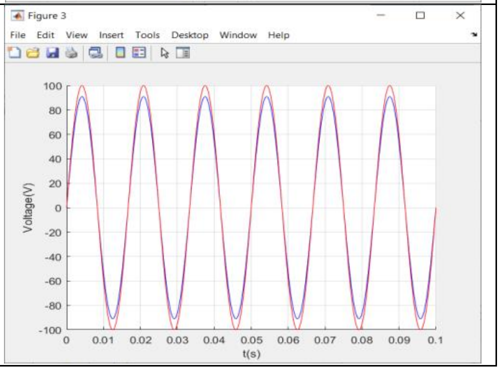
Figure 40. Configuration of the Aurora Link, Port 23.

In order to determine the RSCAD simulator behaviour, the most and least critical k values analysed in ITM Stability and Accuracy Application to DRTS Co-simulation section and different values of timesteps were tested. Table 5 shows the comparison between the theoretical behaviour simulated through a MATLAB model with respect the simulation on RTDS. In all the simulation the Z_A impedance was kept constant at 50 Ω, since from the theoretical analysis it was concluded that what changes the wave behaviour is k. For this reason, by only changing Z_B, k was altered.

In the real-time simulator, when increasing the impedance of Z_B, the system output tends to be less distorted, following in a better way the input signal. Instead when increasing the delay, the opposite happens (Table 5). However, if Z_B is big enough, the output signal (V_A) seems to have an acceptable shape when comparing it to the output waves with smaller timestep. This real time system behaviour is concordant with the theoretical one.

Table 5. Comparison of the theoretical simulation versus the RTDS simulation with different time steps and impedance in Circuit

Timestep (μs)	Z_{B_min}=50.5Ω (RSCAD)	Z_{B_min}=50.5Ω (MATLAB)	Z_B=500 Ω (RSCAD)	Z_B=500 Ω (MATLAB)
500				
250				
100				

50	Subsystem #1 S1) Va S1) E F_Va	Figure 3 Voltage(V) t(s)	Subsystem #1 S1) Va S1) E F_Va	Figure 3 Voltage(V) t(s)
5	Subsystem #1 S1) Va S1) E F_Va	Figure 3 Voltage(V) t(s)	Subsystem #1 S1) Va S1) E F_Va	Figure 3 Voltage(V) t(s)

Focusing on the most critical case which is when Z_B is 50.5 Ω (k=0.9901), the real- time simulator doesn't behave as the theoretical simulation, since the theoretical system stabilizes the wave after the transient, instead the real system doesn't. The wave keeps the initial form the whole time.

When the simulation starts, the transient can be appreciated since the wave does not have still its full amplitude. From Figure 41 it can be seen that at 0.16 s the amplitude of both V_A and I_B stops increasing and its amplitude becomes uniform over time. However, both V_A and V_B continue to be distorted (does not follow completely the shape of a sine wave).

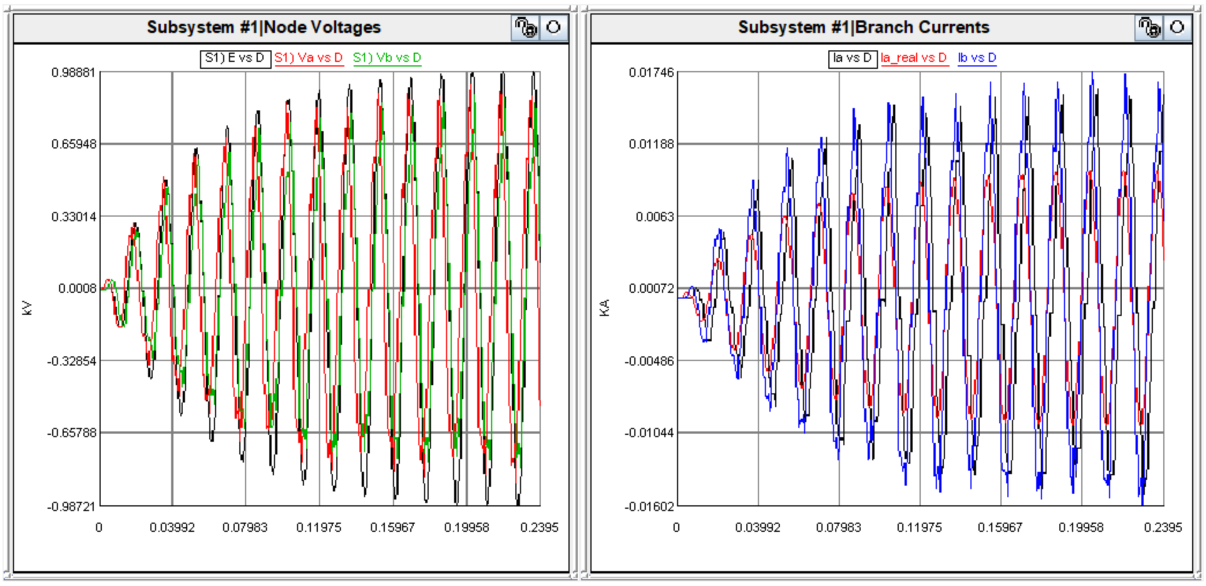


Figure 41. Simulation to determine the amplitude stability of the waves.

Figure 42 shows in a closer view the shape of the waves and the gap between the voltages and currents A and B are appreciated. In both cases, the measured unphased gap varies between 0.0023 and 0.0034 seconds.

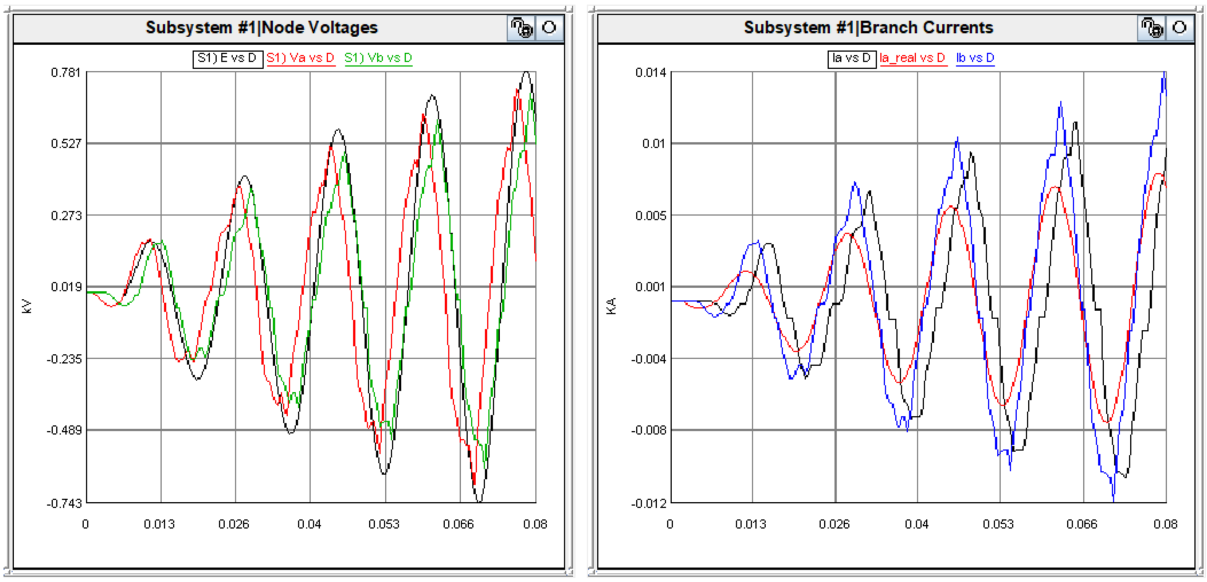


Figure 42. Transient behaviuor when (step time of 500 μs and k=0.9901).

Even though the wave seems to be stable after the first 0.16 s, because keeps following a signal wave pattern over time, it doesn't follow the correct sine wave shape, despite that 10 minutes have passed. This behaviour can be seen in Figure 43.

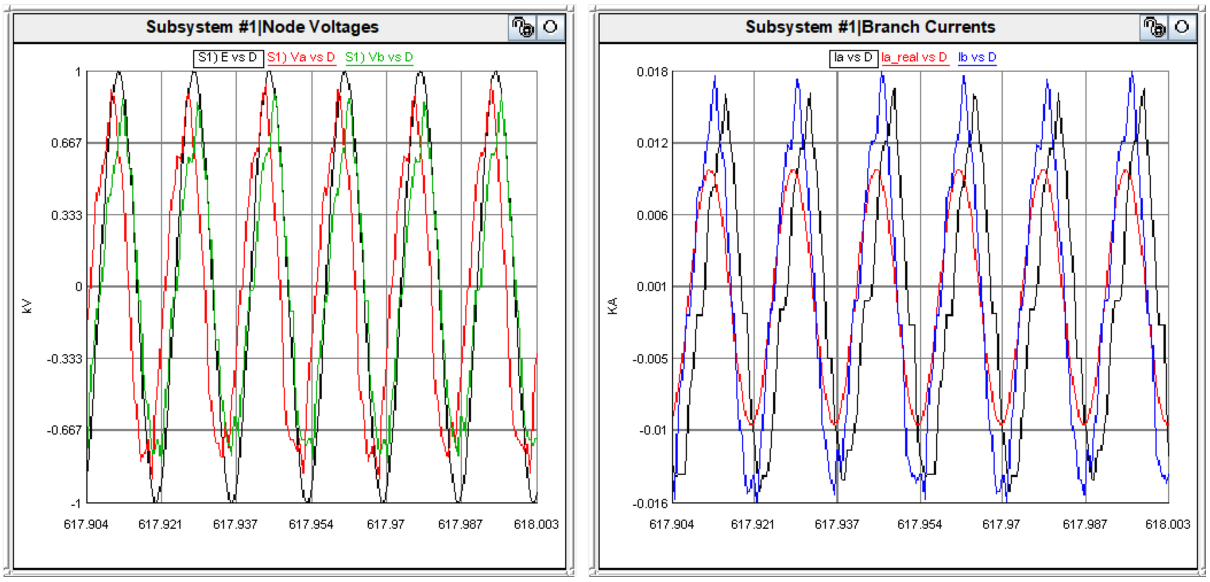


Figure 43. Signal wave shape after 10 min of simulation (step time of 500 µs and k=0.9901).

This result conducts to the conclusion that the simulator is not feasible to perform accurate simulations when two factors are present. The first one is when the parameters that affect stability drives the system too close to the stability limit. The second one is when the step time is too large.

6 Conclusions

This thesis has analysed the problems that arise when performing co-simulation. The main problem is always related with the synchronization of the simulators or platforms, for which this thesis performs an overview of the main co-simulation techniques to reach synchronization. It was also analysed the simulator types, since the appropriate co-simulation technique to implement, depends on the type of interconnected simulators.

The selection of the communication protocol used as platform to transmit and receive information is also an important matter to determine when connecting different physical simulators. In this thesis the properties of the proprietary communication protocols were analysed to determine which to implement in the intended co-simulation. The analysed protocols were: DNP3, Modbus, IEC-104 and Aurora. This last protocol was the one selected to implement in the test cases, because both simulator (RTDS and OPAL-RT) on the Energy Center supported it has no implementation costs.

This thesis aimed to study a co-simulation system when connecting system architecture split parts using the communication protocol Aurora 8B/10B with optical fiber. Looking forward to accomplishing this main objective, the study was divided in two parts. The first part consisted on determining the communication delay when transmitting and receiving information through the same real time simulator (RTS), using the communication protocol Aurora. The second part studies the stability of the co-simulated systems by the deployment of a small electrical system using the Aurora link to connect the two sub-systems, implemented as well using optical fiber in the physical layer.

From the first case, it is concluded that the delay introduced by the use of the communication protocol Aurora (to transmit data from one port to another one) is of 2 times the implemented timestep, regardless of the quantity of transmitted signals or the imposed timestep.

When studying the theoretical approach of the simple test case using the TIM IA approach, the system is stable when k ($k=Z_A/Z_B$) is minor than 1, and unstable for all values of k equal or bigger than 1. In both cases, theoretical and experimental results, when k approaches one (the stability limit), the signals under study (V_A, V_B, I_A and I_B) became more disturbed.

The results obtained from the theoretical and the experimental simulations using the RTDS are similar, which validates the use of TIM IA model. The main encountered difference was that in the theoretical simulations, the output signal reached to stabilize and followed the sine wave shape perfectly after passing the transient time. Instead, in the real-time simulator (RTDS), even though the transient time had passed, the signal was still disturbed and didn't follow the correct sine wave shape.

The previous statements lead to conclude that the simulator is not able to perform accurate simulations when both of the following system configuration settings are present at the same time: *i)* the step time is large (500 microseconds) and *ii)* the parameters that influence the system stability approach the system to the stability limit. So, a compromise must be done when simulating with large time step, since to have accurate results, it is necessary to implement a small k. Acceptable results were found with values of k under 0.2, when imposing a step time of 500 microseconds.

When simulating with small timesteps, all the signals under study, controlled by a remote sent feedback (V_A, V_B, I_A and I_B) stabilized very fast and were always able to follow the correct sine wave shape, for all cases when k was minor than one. Furthermore, the gap between the signals of circuit A and B is very small, which means that the curves V_A and V_B are practically overlap. The same happens in the case of I_A and I_B.

The fact that k must be smaller than one does not imply a direct problem in electrical grid simulation. There is always a natural decoupling (performed by transformers and breakers) between different voltage substations. In particular, when there is a high voltage substation connected, through a transformer, to low/medium voltage substation (a step-down substation), k minor than one is assured (as performed for the test case). Otherwise, when the simulation is intended to be performed in step-up substations (passing form low/medium voltage to high voltage substations) k is bigger than one, thus the system is not stable and co-simulation through the stated methods in this thesis cannot be performed.

Bibliography

[1] T. Yi, L. Feng, W. Qi, C. Bin and N. Ming, "Overview of the Co-simulation Methods for Power and Communication System," in *Coference on Real-Time Computing and Robotics*, Angkor Wat, 2016.

[2] C. Rehtanz and X. Guillaud, "Real-Time and Co-Simulations for the Development of Power System Monitoring, Control and Protection".

[3] W. Ren, M. Steurer and T. L. Baldwin, "Improve Stability and the Accuracy of Power Hardware-in-the-Loop Simulation by Selecting Appropriate Interface Algorithms," *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS,* vol. 44, p. 9, 2008.

[4] Y. Li, D. Shu, F. Shi, Z. Yan, Y. Zhu and N. Tai, "A Multi-Rate Co-Simulation of Combined Phasor-Domain and Time-Domain Model for Large-Scale Wind Farms," *IEEE TRANSACTIONS OF ENERGY CONVERSION,* vol. 35, p. 12, 2020.

[5] C. Yang, Y. Xue, X.-P. Zhang, Y. Zhang and Y. Chen, "Real-Time FPGA-RTDS Co-Simulator for Power Systems," *IEEE,* p. 10, 2018.

[6] W. You, L. Fu and X. Hao, "Cyber-Physical Co-Simulation of Shipboard Integrated Power System Based on Optimized Event-Driven Synchronization," *Electronics (Basel),* vol. 9, 2020.

[7] K. Hopkinsin, IEEE, X. Wang, Member, G. Renan, J. Thorp, K. Birman and D. Coury, "EPOCHS: A platform for Agent-Based Electric Power and Communication Simulation Built From Commercial Off-the.Shelf Components," *IEEE TRANSACTIONS ON POWER SYSTEMS,* vol. 21, 2006.

[8] M. O. Farunque, M. Sloderbeck, M. Steurer and V. Dinavahi, Thermo-Electric Co-Simulation on Geographically Distributed Real-Time Simulators.

[9] B. Chen, A. Goulart and D. Kundur, "Implementing a Real-Time Cyber-Physical System Test Bed in RTDS and OPNET," *IEEE,* p. 6, 2014.

[10] A. Fodor and S. Milán, "Simulation of Electrical Grid with OMNET++ open source discrete event simulator," *Hungarian Journal of Indutry and chemistry,* pp. 85-91, 2016.

[11] NS Network Simulator, "https://www.isi.edu/nsnam/ns/index.html," [Online]. [Accessed 26 01 2020].

[12] J. Martins and M. Portnoi, "TARVOS - an Event Based Simulator for Perfomance Analysis, Supporting MPLS, RSVP-TE, and Fast Recovery," Universidade Salvador, Computer Networks Reasearch, Salvador.

[13] A. S. Sethi and H. Y. Vasil, The Practical OPNET User Guide for Computer Network Simulation, New York: Taylor and Francis Group, LLC, 2011.

[14] OPNET PROJECTS TEAM. CUSTOMIZED, "OPNET Optimum Network Performance," Big Data Projects, 2021. [Online]. Available: https://opnetprojects.com/. [Accessed 26 01 2021].

[15] S. C. Tay, G. Tan and K. Shenoy, "PIGGY-BACKED TIME-STEPPED SIMULATION WITH SUPER STEPPING," *Proceedings of the 2003 Winter Simulation Conference,* 2003.

[16] S. C. Muller, H. Georg, J. Nutaro, E. Widl, Y. Deng and P. Palensky, "Interfacing Power System and ICT simulators: Challenges, State-of-the-Art, and Case Studies," *IEEE Task FOrce Interfacing techniques for Simulation Tools,* pp. 1-11, 2018.

[17] P. Mercier, C. Gagnin, M. Tétreault and M. Toupin, "REAL-TIME DIGITAL SIMULATION OF POWER SYSTEMS AT HYDRO-QUEBEC," *Hydro-Québec Research Institute (IREQ).*

[18] M. McKenna, D. Hamai, M. Kezunovic and Z. Galijasevic, "THE CHOICE OF SIMULATION TIME STEP IN THE REAL-TIME SIMULATOR APPLICATIONS," *Texas A&M University.*

[19] S. Abourida, C. Dufour and J. Bélanger, "Real Time and Hardware-In-The-Loop Simulation of ELectric Drives and Power Electronics: Process, problems and solutions," *The 2005 International Power Electronics Conference,* 2005.

[20] OPAL-RT TECHNOLOGIES, Inc, "OPAL-RT TECHNOLOGIES," [Online]. Available: https://www.opal-rt.com/simulator-platform-op5600/. [Accessed 31 01 2021].

[21] ERIGrid, "ERIGrid," 2020. [Online]. Available: https://erigrid.eu/wp-content/uploads/2018/10/Overview-of-RTDS-Hardware-and-RSCAD-FINAL.pdf. [Accessed 28 October 2020].

[22] RTDS technologies, "RTDS Hardware Manual," RTDS technologies, 2020.

[23] R. Kuffel, J. Giesbrecht, T. Maguire, R. Wierckx and P. MaLaren, "RTDS - A fully Digital Power System Simulator Operting in Real Time," *RTDS Technologies Inc..*

[24] RTDS Technologies Inc., "RTDS Technologies," 2020. [Online]. Available: https://www.rtds.com/wp-content/uploads/2019/08/NovaCor.pdf. [Accessed 28 October 2020].

[25] J. A. Ledin, “Hardware-in-the-Loop Simulation,” *Embedded Systems Programming,* p. 1, 1999.

[26] T. Hatakeyana and A. Riccobono, "Stability and Accuracy Analysis of Power Hardware in the Loop System with Different Interface Algorithms," *IEEE,* p. 8.

[27] J. Siegers and E. Santi, "Improved Power Hardware-in-the-Loop Interface Algorithm Using Wideband Systtem Identification," *IEEE,* p. 7, 2014.

[28] D. Luzeaux and P. Cantot, Simulation and Modeling Systems of Systems, 1st ed., J. W. &. Sons, Ed., London, New Jersey, 2011.

[29] S. J. E. Taylor, "Distributed simulation: state-of-art and potential for operation research," *European Journal of Operational Research 273,* p. 1–19, 2019.

[30] D. L. Mills, COMPUTER NETWROK TIME SYNCRHONIZATION, Florida: Taylor & Francis Group, 2011.

[31] M. Toy, Network and services. Carrier Ethernet, PBT, MPLS-TP, and VPLS, New Jersey: John Wiley &Sons, Inc., Hoboken, New Jersey, 2012.

[32] D. L. Mills, "Network Time Protocol (Version 2)," University of Delaware, Newark, 1989.

[33] M. Ussoli and G. Prytz, "SNTP Time Synchronization Accuracy Measurements," N.D.. [Online]. Available: https://ieeexplore-ieee-org.ezproxy.biblio.polito.it/document/6648120. [Accessed 2020 August 2020].

[34] IEEE STANDARDS ASSOCIATION, "IEEE Standard Profile for Use of IEEE 1588™ Precision Time Protocol in Power System Applications," IEEE-SA Standards Board, New York, 2011.

[35] J.-L. Ferrant, S. Jobert, L. Montini, S. Rodrigues, M. Gilson, M. Mayer, M. Ouellette and S. Ruffini, Synchronous Ethernet and IEEE 1588 in Telecoms, London: WILEY, 2013.

[36] M. Anyaegbu, C.-X. Wang and W. Berrie, "A sample mode packet delay variation filter for IEEE synchronization," *2012 12th Internation Conference on ITS telecomunications,* pp. 1-6, 2012.

[37] M. Han and P. Crossley, "Performance Evaluation of IEEE 1588 for Precision Timing in IEC 61850 Substations," in *Study Committee B5 Colloquium*, Tromsø, Norway, June 24 – 28th 2019.

[38] J. Matson, "Choosing the correct Time Synchronization Protocol and incorporating the 1756-Time module into your Application," Rockwell Automation, Milwaukee, 2013.

[39] L. Dostálek and A. Kabelová, Understanding TCP/IP. A clear and comprehensive guide to TCP/IP protocols, Birmingham: Packt Publishing Ltd. 32, 2006.

[40] S. Kumaran Mani, R. Durairajan, P. Barford and J. Sommers, "A system for Clock Synchronization in an Internet of Things," *Cornell University,* pp. 1-18, 2018.

[41] D. Bailey and E. Wright, "Practical SCADA for Industry," Elsevier, Burlington, 2003.

[42] Modbus Organization, "Modbus," 24 October 2006. [Online]. Available: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. [Accessed 25 July 2020].

[43] MODBUS org, "Modbus," Modbus, 20 December 2006. [Online]. Available: https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf. [Accessed 30 July 2020].

[44] V. Skoko, B. Atlagic and N. Isakov, "Comparative realizaton of IEC 60870-5 industrial protocol standards," 22nd Telecommunications forum TELFOR 2014, Serbia, 2014.

[45] G. Clarke, Modern SCADA Protocols DNP3, IEC 60870.5 and Related Systems, Oxford: Elsevier, 2004.

[46] Xilinix, "Specification, Aurora 8B/10B Protocol," Xilinix, 1 10 2014. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b_protocol_spec_sp002.pdf. [Accessed 12 06 2020].

[47] J.-H. Jung, "Power hardware-in-the-loop simulation (PHILS) of photovoltaic power generaton using real-time simulation techniques and power interfaces," *Jounal of Power Sources,* vol. 285, pp. 137-145, 2015.

[48] D. Shu, X. Xie, Q. Jiang, G. Guo and K. Wang, "A Multirate EMT Co-Simulation of Large AC and MMC-Based MTDC Systems," *IEEE TRANSACTIONS ON POWER SYSTEMS,* vol. 33, p. 12, 2018.

[49] Z. Zhou, Q. Zhang, S. Li and Q. Jin, "A Real-time Co-Simulation Research Based on VSC Closed-Loop Control," *IEEE.*

[50] J. Zhao, H. Wang and H. Zhang, "A Regresion-Based Collaborative Fitlering Recommendation Approach to Time-Stepping Multi-Solver Co-Simulation," *IEEE,* p. 17, 2019.

[51] Q. Wang, W. Tai, Y. Tang, Y. Liang, L. Huang and D. Wang, "Architecture and Application of Real-time Co-simulation Platform for Cyber-physical Power System," in *The 7th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, Hawaii, 2017.

[52] C. Steinbrink, A. A. van der Meer, M. Cvetkovic, D. Babazadeh, S. Rohjans, P. Palensky and S. Lenhoff, "Smart Grid Co-Simulation with MOSAIK and HLA: A Comparison Study," 2007.

[53] M. Platania, "Clock Synchronization," Sapienza University of Rome, Rome.

[54] S. R. Graham, M. E. Coyne, K. M. Hopkinson and S. H. Kurkowski, "A METHODOLOGY FOR UNIT TESTING ACTORS IN PROPRIETARY DISCRETE EVENT BASED," *IEEE,* 2008.

[55] RTDS Technologies, "RTDS Technologies," RTDS Technologies Inc., 2020. [Online]. Available: https://www.rtds.com/technology/communication-protocols/#:~:text=The%20firmware%20supports%20Modbus%20TCP,to%20a%20Modbus%20master%20station.. [Accessed 30 Genuary 2021].

Appendix

APENDIX A. PYTHON CODE TO GENERATE THE .DRAFT AND .SIB DOCUMENTS

```
import time
import datetime
import math
import sys

class Class():
        def __init__(self):
                self.date = datetime.datetime.now()


        def Aurora_blocks(self,disp_max,count_vector,port,blockNum,posx,posy,num_doc):

  ######################################################   TO AURORA
##########################################################################

                dict4="_rtds_Aurora.def\n  {} {} 0 0 521\n  Name:AuroraLnkX\n  Port:{}\n  Proc:1\n
Pri:1\n  sfx:\n  enableSeqNum:No\n  seqNumBlocking:No\n
numVarsToAURORA:$numVar\n".format(posx,posy,port,disp_max)
                print(dict4)
                self.f.write(dict4)
                for i in range(1,129):
                        if i<=disp_max:
                                dict5="  out{}:D{}\n  SetTypeToAURORA{}:Float\n".format(i,i,i)
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(dict5)
                        else:
                                dict6="  out{}:Out{}\n  SetTypeToAURORA{}:Int\n".format(i,i,i)
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(dict6)

  ######################################################   FROM AURORA
##########################################################################
                dict7=f"numVarsFromAURORA:$numVar\n"
                self.f.write(dict7)
                for i in range(1,129):
                        if i<=disp_max:
                                dict5=f"  in{i}:from_aurora{blockNum}_{i}\n
SetTypeFrAURORA{i}:Float\n"
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(dict5)

                        else:
                                dict6=f"  in{i}:in{i}\n  SetTypeFrAURORA{i}:Int\n"
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(dict6)

        def switch_mechanism(self,num_doc):
```

```
                    switch_mechanism="rtds_sharc_ctl_SWITCH\n  432 48 0 0 7\n  Name:SW1\n
Init:On\n  Type:INTEGER\n  VOn:1\n  VOff:0\n  Ton:ON\n  Toff:OFF\nwirelabel\n  464 48 0 0 3\n
Name:SW\n  COL:BLUE\n  Monitor:Yes\nrtds_sharc_ctl_PLOT_UPDATE\n  464 112 0 0 2\n  Proc:1\n
Pri:5\n wirelabel\n  432 112 0 0 3\n  Name:SW\n  COL:BLUE\n  Monitor:Yes\n"
                    self.f = open(f'saved_data{num_doc}.dft','a')
                    self.f.write(switch_mechanism)

          def sib_Documents(self,disp_max,num_doc,aurora_block,SimTime):
                    initial_sib=f"RSCAD 5.011\nINFORMATION FILE: saved_data128.inf\nFILE TO
DOWNLOAD: saved_data128\nFINISH TIME: {SimTime} SECONDS\nPRE-TRIGGER: 20%\nPLOT
DENSITY: EVERY POINT\nMETER UPDATE FREQUENCY: 1000\n"

                    component_switch="COMPONENT: SWITCH\n\
          BOX AT (10,10), DIMENSIONS 60 x 90\n\
          NAME: \n\
          ICON: NONE\n\
          ICON AT: (10,10)\n\
          GROUP: (NONE)\n\
                    GROUP: Subsystem #1|CTLs|Inputs\n\
                    DESC: SW1\n\
          MIN: 0.0\n\
          MAX: 1.0\n\
          ON TEXT: ON\n\
          OFF TEXT: OFF\n\
          TYPE OF DATA: INT\n"

                    component_slider=f"COMPONENT: SLIDER\n\
          BOX AT (792,24), DIMENSIONS 70 x 150\n\
          NAME: \n\
          ICON: NONE\n\
          ICON AT: (10,10)\n\
          GROUP: (NONE)\n\
                    GROUP: DraftVariables\n\
                    DESC: numVar\n\
          UNITS: H\n\
          VALUE: 128\n"

                    componet_plot1="COMPONENT: PLOT\n\
          BOX AT (70,10), DIMENSIONS 536 x 784\n\
          NAME: \n\
          ICON: EXISTS\n\
          ICON AT: (70,10)\n\
          GROUP: (NONE)\n\
          MIN: 0.0\n\
          MAX: 1.0\n\
          UNLOCKED\n\
          NUMBER OF GRAPHS: 3\n\
          X RANGE: DEFAULT\n\
          TIME_ZERO_DESCRIPTOR: Subsystem #1|CTLs|Inputs|SW1\n\
          TIME_ZERO_TRANSITION_TYPE: ANY\n\
          X_DIGITS_OF_PRECISION: 6\n\
X_AXIS_NOTATION: FIXED\n"
        graph1=f"                    GRAPH 1: {disp_max} CURVE\n"
        graf_specs="                            Y MIN: 0.0\n\
                Y MAX: 1.0\n\
                Y LABEL:  \n\
                Y_AXIS_VISIBLE: true\n\
                Y_DIGITS_OF_PRECISION: 6\n\
                MAX_CURVE_LABELS: 6\n\
                X_GRID_LINES: 5\n\
                Y_GRID_LINES: 5\n\
                Y_AXIS_NOTATION: FIXED\n\
                SHOW CURVE LABELS\n\
                SHOW GRID LINES\n\
                SMART SCALE\n"
        graph2=f"                    GRAPH 2: {disp_max} CURVE\n"
        graph3=f"                    GRAPH 3: {disp_max} CURVE\n"

        self.f = open(f'saved_data{num_doc}.sib','a')
        self.f.write(initial_sib)
        self.f.write(component_switch)
        self.f.write(component_slider)
        self.f.write(componet_plot1)
        self.f.write(graph1)
        self.f.write(graf_specs)

        for i in range(1,disp_max+1):
                curve=f"                            CURVE {i}:\n\
                        GROUP: Subsystem #1|CTLs|Vars\n\
                        DESC: D{i}\n\
                        COLOR: BLACK\n\
                        LABEL: D{i}\n"
                self.f = open(f'saved_data{num_doc}.sib','a')
                self.f.write(curve)

        self.f = open(f'saved_data{num_doc}.sib','a')
        self.f.write(graph2)
        self.f.write(graf_specs)
        for i in range(1,disp_max+1):
                        curve=f"                            CURVE {i}:\n\
                        GROUP: Subsystem #1|Aurora\n\
                        DESC: from_aurora1_{i}\n\
                        COLOR: BLACK\n\
                        LABEL: from_aurora1_{i}\n"
                        self.f = open(f'saved_data{num_doc}.sib','a')
                        self.f.write(curve)

        self.f = open(f'saved_data{num_doc}.sib','a')
        self.f.write(graph3)
        self.f.write(graf_specs)

        for i in range(1,disp_max+1):
```

```
                    curve=f"                              CURVE {i}:\n\
                              GROUP: Subsystem #1|Aurora\n\
                              DESC: from_aurora2_{i}\n\
                              COLOR: BLACK\n\
                              LABEL: from_aurora2_{i}\n"

                    self.f = open(f'saved_data{num_doc}.sib','a')
                    self.f.write(curve)

          page_specs="    PAGE HEIGHT: 0.0\n\
     PAGE WIDTH: 0.0\n\
     TOP MARGIN: 0.0\n\
     BOTTOM MARGIN: 0.0\n\
     LEFT MARGIN: 0.0\n\
     RIGHT MARGIN: 0.0\n\
     GRAPH SPACING: 0.0\n"

          self.f = open(f'saved_data{num_doc}.sib','a')
          self.f.write(page_specs)

     def create_Documents(self,disp_max,num_doc,SimTime,DorSib,TimeStep):
          total_components=(disp_max*3)+2
          count=0
          vector_wire=[]
          vector_TimeBlock=[]
          vector_Constant_block=[]
          count_vector=[]
          vector_Capture_block=[]
          vector_Capture_label=[]
          vector_Switch_label=[]
          for x in range(0,15):
                    for y in range(0,10):
                              count=count+1
                              count_vector.append(count)
                              if count>disp_max or count>128:
                                        break
                              else:
                                        x_wire=144+(256*x)
                                        y_wire=400+(96*y)
                                        wirelabel="wirelabel\n  {} {} 0 0 3\n  Name:D{}\n  COL:BLUE
\n  Monitor:Yes\n".format(x_wire,y_wire,count)
                                        vector_wire.append(wirelabel)


                                        TimeBlock="rtds_sharc_ctl_TIME\n  {} {} 0 0 3\n  Mode:time-
step\n  Proc:1\n  Pri:2\n".format(x_wire-32,y_wire)
                                        vector_TimeBlock.append(TimeBlock)


                                        Constant_block="rtds_sharc_ctl_ICONST\n  {} {} 0 0 1\n
Val:0\n".format(x_wire-96,y_wire-32)
                                        vector_Constant_block.append(Constant_block)
```

```
                                                Switch_label="wirelabel\n  {} {} 0 0 3\n  Name:SW\n
COL:BLUE \n  Monitor:Yes\n".format(x_wire-64,y_wire-32)
                                                vector_Switch_label.append(Switch_label)

                                                Capture_block="rtds_risc_ctl_CAPTURE\n  {} {} 0 0 3\n
NP:10\n  Proc:1\n         Pri        :27\n".format(x_wire+32,y_wire)
                                                vector_Capture_block.append(Capture_block)

                                                Capture_label="wirelabel\n  {} {} 0 0 3\n  Name:Out{}\n
COL:BLUE \n  Monitor:Yes\n".format(x_wire+64,y_wire,count)
                                                vector_Capture_label.append(Capture_label)


                var="rtds_draft_var\n\
        432 208 0 0 6\n\
        Name   :numVar\n\
        Type   :INT\n\
        Value  :128\n\
        Max    :128\n\
        Min    :2\n\
        Units  :H\n"

                if int(DorSib)==1:

                        Initial_text="DRAFT 5.011\nTITLE: NumDipositvosUser{}\nCREATED: {} {}, {}
(S264449)\nLAST-MODIFIED: {} {}, {} (S264449)\nTIME-STEP: {}.0E-6\nFINISH-TIME: 0.2\nRTDS-RACK:
1\nCOMPILE-MODE: AUTO\nDISTRIBUTION_MODE: 0\nRTDS REAL-TIME: Yes\nCURRENT-
SUBSYSTEM: 1\nDEFAULT-VIEWMODE: 3\nDEFAULT-ZOOM: 100\nDEFAULT-POINT: 0,0\nDEFAULT-
POINT-ABSOLUTE: 440,298\n1 SUBSYSTEMS\nSUBSYSTEM-TAB-NAME: SS #1\nSUBSYSTEM-CANVAS-
SIZE:3000,2000\nSUBSYSTEM-TITLE:\nSUBSYSTEM-COMMENT:\nSUBSYSTEM-PRINTMODE:    PRINT-
LAYOUT:CUSTOM    PAPER-DIMENSIONS:8.5x11\n{}
COMPONENTS\n".format(disp_max,self.date.strftime("%b"),self.date.strftime("%d"),self.date.strftim
e("%Y"),self.date.strftime("%b"),self.date.strftime("%d"),self.date.strftime("%Y"),TimeStep,total_com
ponents)
                        self.f = open(f'saved_data{num_doc}.dft','a')
                        self.f.write(Initial_text)

                        for x in vector_wire:
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(x)

                        for x in vector_TimeBlock:
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(x)

                        for x in vector_Switch_label:
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(x)


                        self.switch_mechanism(num_doc)
```

```
                                self.Aurora_blocks(disp_max,count_vector,23,1,176,80,num_doc)
                                self.Aurora_blocks(disp_max,count_vector,24,2,176,240,num_doc)
                                self.f = open(f'saved_data{num_doc}.dft','a')
                                self.f.write(var)

                        else:
                                self.sib_Documents(disp_max,num_doc,1,SimTime)


if __name__=="__main__":
        C=Class()
        DorSib=input("Type:\n1 for draft document\n2 for the sib document\n")
        if int(DorSib)==1 or int(DorSib)==2:
                pass
        else:
                print("ERROR! THE PROGRAM WILL FINISH")
                sys.exit()
        input1=input("Type the number of elements (in a range of 2 to 128):\n")
        step=input("Type the step between elements in the document generation (in a range of 1 to
127)\n")
        TimeStep=input("Type the timeStep (units in microseconds):\n")

        Num_steps=2000000

        for i in range(0,int(input1)+1,int(step)):
                if i<2:
                        pass
                else:
                        x=Num_steps*0.000001*float(TimeStep)/int(i)
                        SimTime=round(x,2)
                        C.create_Documents(i,i,SimTime,DorSib,TimeStep)
```

APENDIX B. SIMULATION TIME COMPUTATION

VARIABLES	**TIME STEP (µS)**									
	7	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
2	7.00	10.00	15.00	20.00	25.00	30.00	35.00	40.00	45.00	50.00
3	4.67	6.67	10.00	13.33	16.67	20.00	23.33	26.67	30.00	33.33
4	3.50	5.00	7.50	10.00	12.50	15.00	17.50	20.00	22.50	25.00
5	2.80	4.00	6.00	8.00	10.00	12.00	14.00	16.00	18.00	20.00
6	2.33	3.33	5.00	6.67	8.33	10.00	11.67	13.33	15.00	16.67
7	2.00	2.86	4.29	5.71	7.14	8.57	10.00	11.43	12.86	14.29
8	1.75	2.50	3.75	5.00	6.25	7.50	8.75	10.00	11.25	12.50
9	1.56	2.22	3.33	4.44	5.56	6.67	7.78	8.89	10.00	11.11
10	1.40	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
11	1.27	1.82	2.73	3.64	4.55	5.45	6.36	7.27	8.18	9.09
12	1.17	1.67	2.50	3.33	4.17	5.00	5.83	6.67	7.50	8.33
13	1.08	1.54	2.31	3.08	3.85	4.62	5.38	6.15	6.92	7.69
14	1.00	1.43	2.14	2.86	3.57	4.29	5.00	5.71	6.43	7.14
15	0.93	1.33	2.00	2.67	3.33	4.00	4.67	5.33	6.00	6.67
16	0.88	1.25	1.88	2.50	3.13	3.75	4.38	5.00	5.63	6.25
17	0.82	1.18	1.76	2.35	2.94	3.53	4.12	4.71	5.29	5.88
18	0.78	1.11	1.67	2.22	2.78	3.33	3.89	4.44	5.00	5.56
19	0.74	1.05	1.58	2.11	2.63	3.16	3.68	4.21	4.74	5.26
20	0.70	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
21	0.67	0.95	1.43	1.90	2.38	2.86	3.33	3.81	4.29	4.76
22	0.64	0.91	1.36	1.82	2.27	2.73	3.18	3.64	4.09	4.55
23	0.61	0.87	1.30	1.74	2.17	2.61	3.04	3.48	3.91	4.35
24	0.58	0.83	1.25	1.67	2.08	2.50	2.92	3.33	3.75	4.17
25	0.56	0.80	1.20	1.60	2.00	2.40	2.80	3.20	3.60	4.00
26	0.54	0.77	1.15	1.54	1.92	2.31	2.69	3.08	3.46	3.85
27	0.52	0.74	1.11	1.48	1.85	2.22	2.59	2.96	3.33	3.70
28	0.50	0.71	1.07	1.43	1.79	2.14	2.50	2.86	3.21	3.57
29	0.48	0.69	1.03	1.38	1.72	2.07	2.41	2.76	3.10	3.45
30	0.47	0.67	1.00	1.33	1.67	2.00	2.33	2.67	3.00	3.33
31	0.45	0.65	0.97	1.29	1.61	1.94	2.26	2.58	2.90	3.23
32	0.44	0.63	0.94	1.25	1.56	1.88	2.19	2.50	2.81	3.13
33	0.42	0.61	0.91	1.21	1.52	1.82	2.12	2.42	2.73	3.03
34	0.41	0.59	0.88	1.18	1.47	1.76	2.06	2.35	2.65	2.94
35	0.40	0.57	0.86	1.14	1.43	1.71	2.00	2.29	2.57	2.86
36	0.39	0.56	0.83	1.11	1.39	1.67	1.94	2.22	2.50	2.78
37	0.38	0.54	0.81	1.08	1.35	1.62	1.89	2.16	2.43	2.70
38	0.37	0.53	0.79	1.05	1.32	1.58	1.84	2.11	2.37	2.63
39	0.36	0.51	0.77	1.03	1.28	1.54	1.79	2.05	2.31	2.56
40	0.35	0.50	0.75	1.00	1.25	1.50	1.75	2.00	2.25	2.50
41	0.34	0.49	0.73	0.98	1.22	1.46	1.71	1.95	2.20	2.44
42	0.33	0.48	0.71	0.95	1.19	1.43	1.67	1.90	2.14	2.38
43	0.33	0.47	0.70	0.93	1.16	1.40	1.63	1.86	2.09	2.33

44	0.32	0.45	0.68	0.91	1.14	1.36	1.59	1.82	2.05	2.27
45	0.31	0.44	0.67	0.89	1.11	1.33	1.56	1.78	2.00	2.22
46	0.30	0.43	0.65	0.87	1.09	1.30	1.52	1.74	1.96	2.17
47	0.30	0.43	0.64	0.85	1.06	1.28	1.49	1.70	1.91	2.13
48	0.29	0.42	0.63	0.83	1.04	1.25	1.46	1.67	1.88	2.08
49	0.29	0.41	0.61	0.82	1.02	1.22	1.43	1.63	1.84	2.04
50	0.28	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80	2.00
51	0.27	0.39	0.59	0.78	0.98	1.18	1.37	1.57	1.76	1.96
52	0.27	0.38	0.58	0.77	0.96	1.15	1.35	1.54	1.73	1.92
53	0.26	0.38	0.57	0.75	0.94	1.13	1.32	1.51	1.70	1.89
54	0.26	0.37	0.56	0.74	0.93	1.11	1.30	1.48	1.67	1.85
55	0.25	0.36	0.55	0.73	0.91	1.09	1.27	1.45	1.64	1.82
56	0.25	0.36	0.54	0.71	0.89	1.07	1.25	1.43	1.61	1.79
57	0.25	0.35	0.53	0.70	0.88	1.05	1.23	1.40	1.58	1.75
58	0.24	0.34	0.52	0.69	0.86	1.03	1.21	1.38	1.55	1.72
59	0.24	0.34	0.51	0.68	0.85	1.02	1.19	1.36	1.53	1.69
60	0.23	0.33	0.50	0.67	0.83	1.00	1.17	1.33	1.50	1.67
61	0.23	0.33	0.49	0.66	0.82	0.98	1.15	1.31	1.48	1.64
62	0.23	0.32	0.48	0.65	0.81	0.97	1.13	1.29	1.45	1.61
63	0.22	0.32	0.48	0.63	0.79	0.95	1.11	1.27	1.43	1.59
64	0.22	0.31	0.47	0.63	0.78	0.94	1.09	1.25	1.41	1.56
65	0.22	0.31	0.46	0.62	0.77	0.92	1.08	1.23	1.38	1.54
66	0.21	0.30	0.45	0.61	0.76	0.91	1.06	1.21	1.36	1.52
67	0.21	0.30	0.45	0.60	0.75	0.90	1.04	1.19	1.34	1.49
68	0.21	0.29	0.44	0.59	0.74	0.88	1.03	1.18	1.32	1.47
69	0.20	0.29	0.43	0.58	0.72	0.87	1.01	1.16	1.30	1.45
70	0.20	0.29	0.43	0.57	0.71	0.86	1.00	1.14	1.29	1.43
71	0.20	0.28	0.42	0.56	0.70	0.85	0.99	1.13	1.27	1.41
72	0.19	0.28	0.42	0.56	0.69	0.83	0.97	1.11	1.25	1.39
73	0.19	0.27	0.41	0.55	0.68	0.82	0.96	1.10	1.23	1.37
74	0.19	0.27	0.41	0.54	0.68	0.81	0.95	1.08	1.22	1.35
75	0.19	0.27	0.40	0.53	0.67	0.80	0.93	1.07	1.20	1.33
76	0.18	0.26	0.39	0.53	0.66	0.79	0.92	1.05	1.18	1.32
77	0.18	0.26	0.39	0.52	0.65	0.78	0.91	1.04	1.17	1.30
78	0.18	0.26	0.38	0.51	0.64	0.77	0.90	1.03	1.15	1.28
79	0.18	0.25	0.38	0.51	0.63	0.76	0.89	1.01	1.14	1.27
80	0.18	0.25	0.38	0.50	0.63	0.75	0.88	1.00	1.13	1.25
81	0.17	0.25	0.37	0.49	0.62	0.74	0.86	0.99	1.11	1.23
82	0.17	0.24	0.37	0.49	0.61	0.73	0.85	0.98	1.10	1.22
83	0.17	0.24	0.36	0.48	0.60	0.72	0.84	0.96	1.08	1.20
84	0.17	0.24	0.36	0.48	0.60	0.71	0.83	0.95	1.07	1.19
85	0.16	0.24	0.35	0.47	0.59	0.71	0.82	0.94	1.06	1.18
86	0.16	0.23	0.35	0.47	0.58	0.70	0.81	0.93	1.05	1.16
87	0.16	0.23	0.34	0.46	0.57	0.69	0.80	0.92	1.03	1.15
88	0.16	0.23	0.34	0.45	0.57	0.68	0.80	0.91	1.02	1.14
89	0.16	0.22	0.34	0.45	0.56	0.67	0.79	0.90	1.01	1.12

90	0.16	0.22	0.33	0.44	0.56	0.67	0.78	0.89	1.00	1.11
91	0.15	0.22	0.33	0.44	0.55	0.66	0.77	0.88	0.99	1.10
92	0.15	0.22	0.33	0.43	0.54	0.65	0.76	0.87	0.98	1.09
93	0.15	0.22	0.32	0.43	0.54	0.65	0.75	0.86	0.97	1.08
94	0.15	0.21	0.32	0.43	0.53	0.64	0.74	0.85	0.96	1.06
95	0.15	0.21	0.32	0.42	0.53	0.63	0.74	0.84	0.95	1.05
96	0.15	0.21	0.31	0.42	0.52	0.63	0.73	0.83	0.94	1.04
97	0.14	0.21	0.31	0.41	0.52	0.62	0.72	0.82	0.93	1.03
98	0.14	0.20	0.31	0.41	0.51	0.61	0.71	0.82	0.92	1.02
99	0.14	0.20	0.30	0.40	0.51	0.61	0.71	0.81	0.91	1.01
100	0.14	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
101	0.14	0.20	0.30	0.40	0.50	0.59	0.69	0.79	0.89	0.99
102	0.14	0.20	0.29	0.39	0.49	0.59	0.69	0.78	0.88	0.98
103	0.14	0.19	0.29	0.39	0.49	0.58	0.68	0.78	0.87	0.97
104	0.13	0.19	0.29	0.38	0.48	0.58	0.67	0.77	0.87	0.96
105	0.13	0.19	0.29	0.38	0.48	0.57	0.67	0.76	0.86	0.95
106	0.13	0.19	0.28	0.38	0.47	0.57	0.66	0.75	0.85	0.94
107	0.13	0.19	0.28	0.37	0.47	0.56	0.65	0.75	0.84	0.93
108	0.13	0.19	0.28	0.37	0.46	0.56	0.65	0.74	0.83	0.93
109	0.13	0.18	0.28	0.37	0.46	0.55	0.64	0.73	0.83	0.92
110	0.13	0.18	0.27	0.36	0.45	0.55	0.64	0.73	0.82	0.91
111	0.13	0.18	0.27	0.36	0.45	0.54	0.63	0.72	0.81	0.90
112	0.13	0.18	0.27	0.36	0.45	0.54	0.63	0.71	0.80	0.89
113	0.12	0.18	0.27	0.35	0.44	0.53	0.62	0.71	0.80	0.88
114	0.12	0.18	0.26	0.35	0.44	0.53	0.61	0.70	0.79	0.88
115	0.12	0.17	0.26	0.35	0.43	0.52	0.61	0.70	0.78	0.87
116	0.12	0.17	0.26	0.34	0.43	0.52	0.60	0.69	0.78	0.86
117	0.12	0.17	0.26	0.34	0.43	0.51	0.60	0.68	0.77	0.85
118	0.12	0.17	0.25	0.34	0.42	0.51	0.59	0.68	0.76	0.85
119	0.12	0.17	0.25	0.34	0.42	0.50	0.59	0.67	0.76	0.84
120	0.12	0.17	0.25	0.33	0.42	0.50	0.58	0.67	0.75	0.83
121	0.12	0.17	0.25	0.33	0.41	0.50	0.58	0.66	0.74	0.83
122	0.11	0.16	0.25	0.33	0.41	0.49	0.57	0.66	0.74	0.82
123	0.11	0.16	0.24	0.33	0.41	0.49	0.57	0.65	0.73	0.81
124	0.11	0.16	0.24	0.32	0.40	0.48	0.56	0.65	0.73	0.81
125	0.11	0.16	0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80
126	0.11	0.16	0.24	0.32	0.40	0.48	0.56	0.63	0.71	0.79
127	0.11	0.16	0.24	0.31	0.39	0.47	0.55	0.63	0.71	0.79
128	0.12	0.16	0.23	0.31	0.39	0.47	0.55	0.63	0.70	0.78