FACHHOCHSCHUL - MASTERSTUDIENGANG

AUTOMATISIERUNGSTECHNIK

Expanding data exchange capabilities for RTDS using an industry based communication protocol

RESEARCH PAPER

by

Andreas Berger

20. March – 25. September 2013

Assistance of the Research by

Prof. (FH) Univ. Doz. Dipl.-Ing. Dr. Karl Heinz Kellermayr
Dr. Lukas Graber

ACKNOWLEDGEMENT

The Center for Advanced Power Systems (CAPS) is a state-of-the-art research facility of the Florida State University and possesses the reputation in doing outstanding research in the field of power systems technology. I have learned a lot throughout my stay at CAPS, with many challenging yet valuable experiences in order to complete this work. This would not have been possible without the help of many people why I want to use this portion of my thesis to mention all those who helped me with my research project. First I would like to thank Dr. Michael „Mischa“ Steuer who has given me the opportunity to accomplish my project in the Power Systems Group at CAPS and without whom this thesis would not have been possible at all. I would also like to express my gratitude to my supervisor Dr. Lukas Graber who helped me throughout the process obtaining my project and who made working on this thesis very manageable and a great learning experience due to his congenial and easily approachable nature, along with being a pool of knowledge. Thanks also goes to my supervisor in Austria Prof. (FH) Univ. Doz. Dipl.-Ing. Dr. Karl Heinz Kellermayr from University of Applied Sciences Upper Austria who was very supportive, helpful and guided me really well from the beginning of this project until its completion beside from CAPS. I benefited significantly from the feedback and suggestions of my supervisors not only in technical terms but also in regard to preparing this paper. I would also like to thank Mr. Mike Sloderbeck for his friendly support and for helping me to understand the FPGA evaluations boards and the RTDS interface.

I also want to thank Dipl.-Kulturw. Vanessa Prüller from the international office at my home university for her support with the international affairs to enable the accomplishment of the internship at CAPS. Further, I want to thank the Austrian Marshallplan Foundation as I did the research for this Master Thesis as a scholar from the Austrian Marshallplan Foundation. Last but not least, I would like to thank my family for supporting me during my presence as a student and everyone else who has been involved in my projects in one way or another.

ABSTRACT

The research presented in this thesis has been performed in 2013 at the Center for Advanced Power Systems (CAPS) of Florida State University. The Center for Advanced Power Systems is a research center of the Florida State University (FSU) with the focus on control systems, power electronics and machines, thermal management, high temperature superconductor characterization and electrical insulation research. Because of tis multidisciplinary research, CAPS is a well-known research institution with an outstanding reputation in the field of control systems and power electronics and machines. Focus of Research is the All Electrical Ship concept of the US Navy. Extensive simulation studies are performed on this topic at CAPS and an important tool for this work is a Real Time Digital Simulator (RTDS).

Topic of research in this thesis is the concept of expanding data exchange capabilities for Real Time Digital Simulator. In chapter 1 a short introduction to the future All Electrical Ship (AES) of the U.S. Navy concept and the topic of power system grounding are given. For research in that topic, extensive simulation studies are of great importance. Powerful Simulation tools are needed. Chapter 2 describes the RTDS and its interface to external equipment (HIL - Hardware in the Loop simulation) via analog to digital and digital to analog connectors. To expand the possibilities of RTDS, a project was set up to create a digital method of communication with an industry standard protocol which offers multi-channel possibilities, low-latency and high-throughput. It is desirable to improve on analog measurements and control signals. This will facilitate for higher flexibility in Controller-Hardware-In-the-Loop (CHIL) based simulations. Chapter 3 and 4 show and explain in detail the implementation and program of the selected Aurora 8B/10B protocol on a Field Programmable Gate Array (FPGA). The Aurora protocol, developed by Xilinx, is a high performance (many gigabits/second) scalable, lightweight, link-layer protocol that is used to move data across point-to-point serial links. It is an open protocol and is free of charge. It provides a transparent interface to the physical serial links, allowing upper layers of proprietary or industry-standard protocols to easily use these high-speed serial links. Aurora is an efficient low-latency protocol that uses the least possible amount of logic while offering a rich, highly configurable feature set. Aurora increases bandwidth and throughput

through bonded lanes. The protocol was verified in a CHIL experiment which uses a custom controller board for voltage source conversion and novel design emphasis flexibility and is described in chapter 5. This custom board allows fiber optic communication between subsystems and was chosen because of this purpose.

EXECUTIVE SUMMARY

This master thesis is result of an internship at the Center for Advanced Power Systems (CAPS). CAPS is a research facility of Florida State University (FSU) and one of the major partners in the Electric Ship Research and Development Consortium (ESRDC) of the United States Office of Naval Research (ONR). The focus of the research described in this thesis is on the topic of expanding interface possibilities of the Real Time Digital Simulator (RTDS) to facilitate for higher flexibility in Controller Hardware In the Loop (CHIL) based simulations, such as ongoing research projects in the ESRDC with the aim to support the development of next-generation all-electric naval ships. These refer to ships whose power generation, propulsion and auxiliary systems entirely depend on electric power. Further, these ships employ new ship designs and have the ability to make use of advanced weaponry. In recent years, the complexity of power systems and hardware interfaces have been growing, making Hardware-In-the-Loop (HIL) simulation more difficult for systems with different data exchange methods. One of the challenges in simulating complex power systems is to guarantee faultless communication and high data throughput between components. The RTDS interfaces to external equipment generally via analog to digital and digital to analog connectors. It is desirable to improve on analog measurements and control signals, to expand the possibilities of the RTDS.

This thesis introduces the concept of a high speed digital interface between the real-time simulator and other controllers, hardware or even other simulators. In the digital interface, the communication between the platforms is achieved through the reconfigurable Field Programmable Gate Array (FPGA) boards. A high speed serial protocol was used to create a digital method of communication with an industry standard based protocol. This one offers multi-channel possibilities, low-latency and high-throughput and should stand for a new possibility and standard for connections between simulators and hardware. *The goal of the thesis is to expand data exchange capabilities for RTDS using an industry based communication protocol.*

The protocol that has been selected is the Aurora protocol, developed by Xilinx. Aurora is a scalable, lightweight, link-layer protocol that is used to move data across point-to-

point high speed serial links. It is an open source protocol and is free of charge. It provides a transparent interface to the physical serial links, allowing upper layers of proprietary or industry-standard protocols to easily use these high-speed serial links. Aurora is an efficient low-latency protocol that uses the least possible amount of logic while offering a rich, highly configurable feature set. Aurora increases bandwidth and throughput through bonded lanes. These interfaces also leverage the strength of each simulation platform and can be used to interface the controller with a real-time simulator, thus enhancing the Controller-Hardware-In-the-Loop (CHIL) and general Hardware-In-the-Loop (HIL) capabilities of the real-time simulators.

The protocol is implemented on a FPGA evaluation board and verified in a CHIL experiment which uses a custom controller board for voltage source conversion and novel design emphasis flexibility. This custom board allows fiber optic communication between subsystems and was chosen because of this purpose.

INDEX

1 ALL-ELECTRIC-SHIP CONCEPT FOR FUTURE'S NAVAL VESSELS

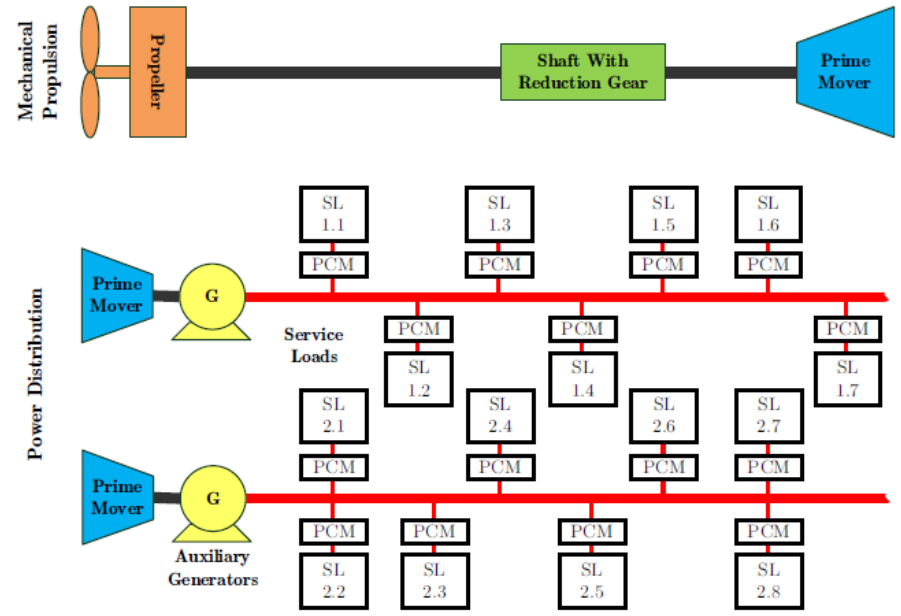
1.1 ALL-ELECTRIC-SHIP (AES)

In recent years, the United States Navy began to reconsider the concept of traditional naval ship systems as a consequence of growing costs and military capabilities with the goal to make ships more efficient and affordable by exploring new and more reconfigurable system architectures. Conventional naval ships employ geared mechanical drive systems for propulsion. These ships host two separate sets of engines to provide the necessary energy for ship propulsion and service loads [1]. One set of engines is directly-linked to the mechanical propulsion system since a large amount of installed power is dedicated to the propulsion [2]. Typically this set contains two pairs of gas turbines as the prime movers whereby each pair is connected to a large common reduction gearbox to reduce the engine's high rotational-speed and to permit the operation of the propeller at lower rotational speeds. For survivability reasons one drive train must be located forward and the other aft. As a result, long line shafts of the propulsion system are running through the ship to connect the gearboxes and rotate the propellers. Furthermore, the propellers location defines the centerline of the shaft lines and hence the vertical position of the prime movers. The required location as well as the size and number of the installed machinery of the mechanical propulsion system are considerable factors which reduce the internal space available for equipment other than propulsion, prohibit efficient use of capacity, and restrict ship designers in their design flexibility. It might be fair to say the rest of the ship needs to be arranged and designed around these set of engines instead of creating a customized propulsion system for the ship. A second set, typically including three gas turbines, is connected to smaller auxiliary generators to operate all the electrically powered equipment on these ships. It is a fact, that naval warships operate relatively little time at maximum speed [3].The majority of the installed power generation is locked in the propulsion system and not available for non-propulsion loads. The overall efficiency of these ships can suffer due to the inefficiency of the propulsion prime movers at low operation speed [3]. These facts have an impact to increased fuel consumption, increased operation costs, system

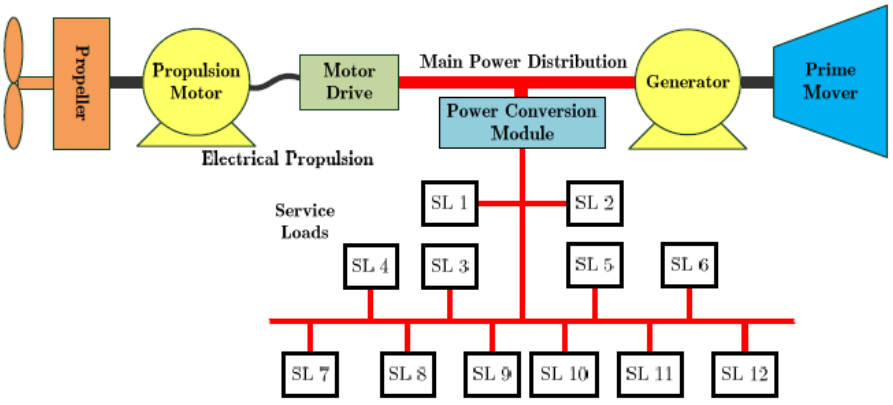
efficiency and mission flexibility. Over the recent decades electric-drive technology has been considered and introduced and finally accepted as possible alternative to mechanical drive systems. With the beginning of the eighties, ships built with electrical propulsion systems have become more popular and are found today in many types of ships such as cruise liners, icebreakers and amphibious assault ships. Electric propulsion systems eliminate the need of a mechanical link between prime movers and propeller. Propulsion shafts can be shorter and do not need to be aligned any longer [4]. The missing physical connection enables cross-connecting between power generation and power operated ship systems. This provides ship designers a considerable amount of arrangement flexibility to place the gas turbines where they will optimize space, weight and balance of the ship without interfering with ship mission equipment. The introduction of electric propulsion equipment into naval ships resulted in a considerable increase of the electric power level, since usually the electric propulsion represents the most demanding electrical load on the ship [5]. Furthermore, in future the US Navy is moving towards more unconventional mission weapon systems, weapon launchers and sensors with higher electrical power requirements such as high energy Laser weapon, electro-magnetic gun, electro-magnetic assisted aircraft launcher or Radar systems. They are gaining increased importance due to change in the philosophy of naval warfare. To meet the requirements of changing trends and demands regarding to power, improved mission capabilities along with superior system efficiencies and decreased operating costs in future, the U.S. Navy is investing substantial in exploring ways to power so-called all-electric ships (AES) [1]. This next-generation ship concept is being mapped out in a program by the U.S. Office of Naval Research known as the Next-Generation Integrated Power System (NGIPS). It aims at to make more efficient use of on-board power by incorporating the knowledge learned from utilization and development of the Integrated Power System (IPS) which has been developed by the U.S. Navy between 1992 and 2006 [6]. Other than traditional warships the AES refers to a ship whose power generation, propulsion and auxiliary systems entirely depend on electric power (as opposed to hydraulic, steam or compressed-air powered systems). The IPS is evolved as to be the most appropriate solution to overcome the increasing power demand and to fulfill all electrical needs of a future naval warship [7]. Unlike traditional power system configurations, which have dedicated power sources for propulsion and ship electrical power generation (also known as Segregated Power

System), the IPS architecture features electric-drive propulsion and is designed to produce power for both propulsion and ship service systems and enables the development of affordable configurations for a wide range of ship applications such as submarines, surface combatants, aircraft carriers, amphibious ships, auxiliary ships, sealift and high value commercial ships by providing a common pool of electrical power throughout the ship. Combing propulsion and electrical ship service system into a single power system not only reduces the total number of prime movers but also provides a considerable increase of overall system efficiency compared to an equivalent mechanical drive design [3]. All prime movers in an IPS are coupled to generators which are capable to produce more than the maximum peak power for the propulsion but less than the sum of peak power of all electric loads. On top of this the IPS allows to release and reallocate the large amount of power which is dedicated to the propulsion to accommodate combat systems for any given operating condition, especially when high-speed propulsion is not an operational requirement. The IPS architecture can be implemented in Zonal Power Distribution (ZED) structure [8]. The design method of dividing the ship into multiple electrical zones allows isolating each zone from each other and enables the ship to withstand a certain level of damage. On the power-level, prospective electrical faults and their propagation can be localized and controlled in a more efficient fashion. Loads outside two adjacent damaged zones do not see an interruption of power. [9]. This added robustness together with the ability that electrical power (contrary to mechanical power) can be relatively easily redistributed to loads of higher priority in battle mode or, in case that damage occurs on the ship, to undamaged propulsion systems, service or mission critical combat systems. This results in a substantially increased flexibility in ship design re-configurability and combat survivability. Although the advances in ship technologies and the scheme of power design offers anticipated benefits in many respects and provides ship designers opportunities to meet the individual needs and operational requirements of different ship types far easier, this architecture makes the shipboard grid structure rather complex due to multiple interacting (sub)-systems. Impacts on the ship equipment are more serious in case a fault occurs as a consequence of higher voltage levels [7]. Electric-drive propulsion equipment might be larger and heavier compared to equivalent mechanical-drive devices. As with any industrial product, the component-level technology is moving towards becoming smaller and lighter by being more energy efficient. It is

expected that size and weight of electric-drive systems will come down over time to dimensions comparable to mechanical drive systems. Nevertheless the merits of the integrated electric-drive propulsion such as fuel savings can offset the size and weight drawback already today [1].



(a) Conventional Ship Design showing prime movers, auxiliary generators, service loads (SL), and power converter modules (PCM)



(b) Electric Propulsion with Integrated Power

Figure 1.1: Simplified shipboard power system comparing a conventional design [4].

Figure 1.1 compares a conventional ship design with a design based on IPS. Although this figure does not correspond to any real ship's one-line diagram (multiple generators, motors and conduction paths for reliability and survivability), it provides a basic

understanding of the essential differences of a power system configurations including mechanical ship propulsion compared to an integrated electric architecture. Even if commercial ships are already using electric propulsion the concept and development of a next generation naval ships based on AES is still in its infancy [1].

The utilization of large power level generation and distribution on ships with their severe space constraints, poses new multidisciplinary challenges in ship design and manufacturing. As there are many uncertain parameters in many respects which need to be addressed, the United States ONR[1] established the Electric Ship Research and Development Consortium[2] (ESRDC). ESRDC is a consortium of eight individual research institutions with fundamental knowledge in the field of power systems technology and focuses on research in electric ship concepts. These institutions are dedicated not only to furthering research in electric ship and other advanced electric technologies, but also to address the national shortage of electric power engineers by providing educational opportunities [14].

1.2 Electric Power Distribution Architectures For Future Naval Ships

The selection of power system architecture for a ship primarily depends on the specific requirements of the ship under design and attached to this on factors such as quality, efficiency, survivability and reliability desired. Currently three main systems are being considered which may have application in different future ship types associated to the US Navy: Medium-Voltage AC (MVAC), High-Frequency AC (HFAC) and Medium-Voltage DC (MVDC). Whereas the first two architectures are proposed for near term applications, is the Medium-Voltage DC (MVDC) considered to be the most appropriate power system architecture for future's naval all-electric warships (see Fig. 1.2).

[1] http://www.onr.navy.mil/
[2] http://www.esrdc.com/

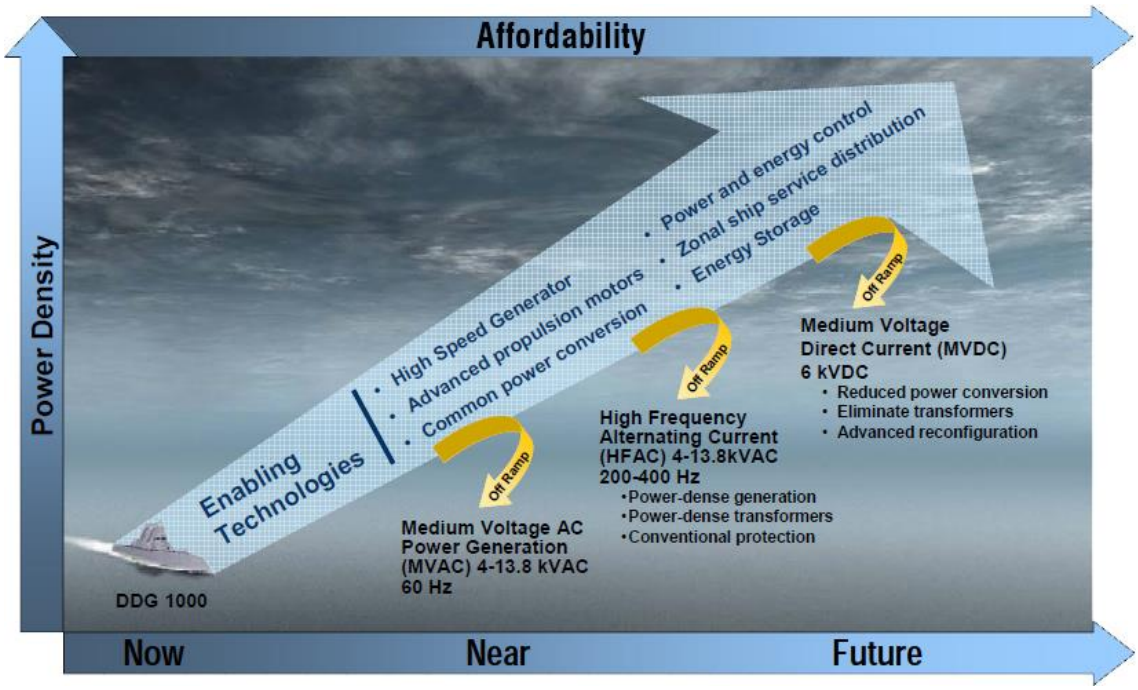


Figure 1.2: Next Generation Integrated Power System (NGIPS) [10]

Medium-Voltage AC (MVAC): In an MVAC integrated power system, power is generated and shared by means of a 3-phase distribution bus at conventional power frequency of 60 Hz at one of three standard voltage levels (4:16 kV, 6:9 kV or 13:8 kV). The MVAC architecture is appropriate for ships where high power density is not a requirement [10]. The MVAC architecture is well investigated by the shipbuilding community induced due to the similarity to land-based (i.e. terrestrial) AC power distribution systems and thus can be deployed in the ship with limited development effort.

High-Frequency AC (HFAC): Unlike the MVAC integrated power system, power is generated at a fixed frequency at or above 200 Hz rather than 60 Hz but less than 400 Hz and hence referred to as the high frequency architecture. HFAC is an intermediate step towards the MVDC architecture and for ships with high power density requirements. On one hand it provides several advantages compared to MVAC such as a reduction of size and weight of transformers and rotating machinery, improved acoustic performance or lower harmonic distortion. On the other hand, the higher

frequency also introduces several challenges such as high number of poles being required for generators or prime movers are restricted to specific operating speeds [10].

Medium-Voltage DC (MVDC): The MVDC integrated power system is the target NGIPS architecture to meet the high power density requirements of future surface combatants and submarines. The primary difference on system-level compared to an (HF)AC power system is that power is generated as AC, but distributed as DC power through out the ship. Consequently a couple of conceptual features come along with this architecture. The prime movers speed is fully decoupled from the frequency of the main distribution bus which enables the possibilty to optimize the generators for each type of prime movers without gears. Engineering concerns (e.g. cable installations) with respect to electromagnetic interference (EMI) and electromagnetic compatibility (EMC) present with HFAC architecture are reduced. On system-level the DC system is simpler, since many conversion steps are avoided. Each step conversion causes losses and potentially reduces reliability of the system. Overall the MVDC architecture is predicted to provide the highest payoff by promising highest capabilities at the least total cost of ownership. However, the application and conceptual changes of the MVDC architecture make this the technically most challenging one of all three architectures defined [10].

The pros and cons of the integrated power system architectures are described more detailed in [10].

The architectures mentioned above have in common, that the choice of appropriate grounding schemes is of utmost importance for the performance of the power system [14].

1.3 SHIPBOARD GROUNDING SCHEMES

The grounding strategy of an electrical shipboard power distribution system is a major aspect since the primary objective is personel safety, availability and continuity (reliability) of the electrical power supply in case of a system fault. Traditionally, literature and especially established engineering practices, distinguish between ungrounded, impedance grounded (high or low impedance) and effectively grounded power systems. Strictly speaking there are no ungrounded systems since there is always

some degree of parasitic coupling to ground present. Therefore, ungrounded systems are also understood as parasitically or not intentionally grounded systems. For ships the unintentionally or high-impedance grounding schemes are preferred since those allow to continue operation of all equipment throughout the ship in case of a single-rail-to ground fault (without clearing the first ground fault exists) [11]. But these grounding schemes come with their own set of issues. One problematic issue arises when locating such a fault since those do not provide considerable fault currents for tracing fault location. Capacitive, resistive, and inductive grounding have specific characteristics which need to be carefully studied. Also the location of the engineered grounding point has substantial impact to the performance of the power system during normal operation and during faults. Despite the importance of the engineered grounding for shipboard electrical power systems the established practices [12] still rely on rather simplified analytical approaches which have primarily been established and validated for terrestrial power systems [13]. While the grounding practices of AC shipboard power systems are generally well resolved, there is very little literature available and even less guidance given how to engineer grounding for DC power systems on ships [11].

Therefore, the focus research in the field of power systems grounding conducted by the Electric Ship research and development Consortium (ESRDC) is on grounding aspects of direct current (DC) systems, especially operated at medium voltage (MVDC) [14].

2 REAL TIME DIGITAL SIMULATION (RTDS)

This chapter describes details on the design, architectural features and applications of a real-time digital simulator (RTDS). The combination of real-time operation, flexible I/O, Graphical-User-Interface (GUI) and an extensive library of accurate power system component models make the RTDS an ideal simulation tool with a wide range of applications.

2.1 INTRODUCTION

Simulation has long been recognized as an important and necessary step in the development, design and testing of power generation and transmission systems. A wide variety of both analogue and digital simulation tools are available and typically used during various stages of system development. Recent advances in both computing hardware and sophisticated power system component modeling techniques have significantly increased the application of digital simulation in the power system industry. Of particular interest in the context of this paper are the advances made in the study of electromagnetic transients' phenomenon. The study of power system transients deals with events which occur due to a disturbance on the power system (e.g. line faults, breaker switching, etc.). Under such disturbed conditions the power system and the components within it are subjected to stresses which far exceed those experienced during steady state operation. Transient simulations are performed in order to examine these stresses and to evaluate what effects they will have on the system being studied. Results obtained from transient simulations can be used by engineers to ensure that component ratings are sufficient and that appropriate protective devices are in place to prevent damage to components when faults occur. The traditional method for performing transient studies utilize simulators made up of scaled down power system components. Each component is physically connected to the next in a manner similar to that in the real system. This analogue simulation technique forms the basis of both the Transient Network Analyses (TNA) and the HVDC simulator. The second method commonly applied in the study of electromagnetic transient phenomenon is based on a mathematical representation of the system and its dynamics rather than on scaled down physical components [15] [16] [17].

One of the most significant disadvantages of software based simulators relates to the speed at which they operate. Unlike analogue simulators which operate in real time, most digital simulation systems operate in non-real time. Real time operation implies that an event in the system which lasts for one second can be simulated on the simulator in one second Digital simulations typically quire many seconds or minutes to perform the necessary computations and produce a solution for such an event. This so-called non-real time operation often limits the application of digital simulators, particularly when testing of physical control and protection equipment is being considered [20].

2.1.1 REAL TIME DIGITAL SIMULATION

Recent advances in digital signal processing, together with improvements in the accuracy and efficiency of power system component models, have had a significant effect on digital simulation technology. It has been demonstrated that for relatively simple systems, real time operation can be achieved for short periods of time using high end, high speed computer workstations [18][19]. However, as the size of the systems modeled increases, the number of computations which must be performed on the single CPU rises quickly. Although these developments are significant and encouraging, their application is still somewhat limited.

An alternative approach to achieving real time operation was investigated and implemented at the Manitoba HVDC Research Centre (Winnipeg, Canada). In **this** approach, many digital signal processors share the computational burden required to solve the emtp equations and algorithms. Because of its parallel nature, increases in system size can be accommodated by increasing the number of processing elements. In this way, the represented power system can grow without affecting the real time capability of the digital simulator. The RTDS is a combination of specialized computer hardware and software designed specifically for the solution of power system electromagnetic transients. An extensive power system component library together with a friendly graphical user interface, facilitate the assembly and study of a wide variety of ac, dc and integrated ac/dc power systems. Because real time operation can be achieved and in definitely sustained, the RTDS can be applied in areas traditionally reserved for analogue simulators [20].

2.2 RTDS HARDWARE

The RTDS is organized into individual racks of tightly coupled digital signal processors connected to one another using a common backplane (see Fig. 2.1). Each rack is identical and contains three distinct types of cards:

- Tandem Processor Card (TPC)

-Workstation Interface Card (WIC)

- Inter-Rack Communication Card (IRC)

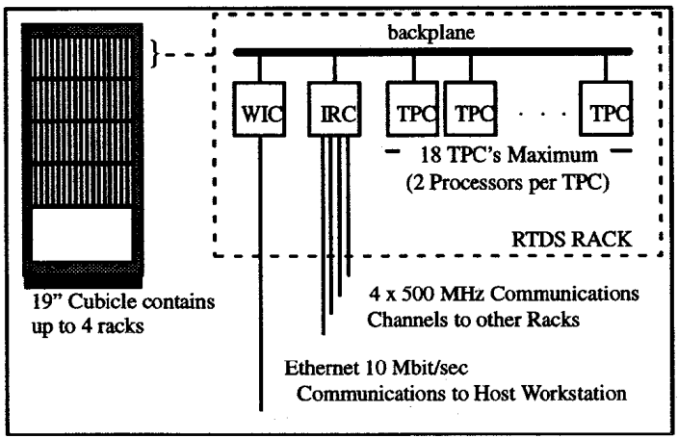


Figure 2.1: RTDS Hardware Architecture [20]

The real-time digital simulator at CAPS is a parallel RTDS power-system simulator containing nearly 500 processors [20]. The RTDS is designed to simulate systems with time-step sizes on the order of 50 μs. For power electronics, the RTDS provides a feature to simulate such subsystems with smaller time-steps (typically 2 μs). The simulator at CAPS has 330 Analog Devices ADSP21062 (SHARC) digital signal processors and 50 RISC processors (IBM PPC750CXe and IBM PPC750GX), operated in parallel, and provides digital and analog I/O ports for interfacing hardware to the simulation. The system is scalable, allowing subsystems of up to 54 electrically accessible nodes to be simulated on a single "rack", while larger systems can be

simulated by connecting together subsystems simulated on separate racks. There are 14 RTDS racks installed at the real-time power systems simulation facility at CAPS. The RTDS can simulate both electrical networks and control systems in real time, meaning one 50 μs time step of the simulation takes exactly 50 μs of real-world time to execute and thus the evolution of the simulation is synchronized precisely with the evolution of real-world hardware dynamics. The RTDS analog input ports are connected to measurement probes and to the signal outputs of the power systems equipment in the CAPS facility, and the RTDS analog output ports provide reference values to the equipment controllers in the facility, which are described below [25].

2.2.1 TANDEM PROCESSOR CARD

Eighteen tandem processors (TPCs) reside in each RTDS rack. Each TPC contains two digital signal processors with a combined computing speed of approximately 44 MFLOPS (millions of floating point operations per second). Since each TPC is identical, its function in any simulation case is purely defined by software. Depending on the type of function allocated to a particular TPC during a simulation, its two processors may operate independently or in unison. This flexibility provides increased processing power for component models with more complicated algorithms. High speed data exchange can be achieved between adjacent processors without having to access the interconnection backplane. In order to minimize internal data communication and to facilitate interconnection to the RTDS to measurement, control and protection devices, each TPC is provided with analogue and digital I/O. In general, half of the I/O on each TPC is assigned to each processor so that locally computed quantities can be directly output (input). Under certain circumstances however, since the system is fully programmable, signals can be passed to output ports belonging to the adjacent processor or to ports on other TPCs. Each TPC contains the following I/O:

- eight scaleable analogue output ports

- two 16-bit digital output ports

- two 16-bit digital input ports

- two analogue input ports (optional)

The actual function allocated to an I/O port will vary depending on the particular model assigned to the processor during any given simulation [20].

2.2.2 WORKSTATION INTERFACE CARD

Each RTDS rack also contains one workstation interface card. The main task of the WIC is to regulate communication both between the Tpcs which reside on the particular rack and between the rack and the host computer workstation. RTDS simulation cases are run using sophisticated graphical interface software installed on the host computer. Connection between the RTDS rack(s) and the workstation is via standard ethemet based local area network. An ethemet controller on the WIC is able to interpret data packets which are intended for its use and respond back to the originating workstation (based on unique addresses which are assigned to each rack and workstation). Incoming data can be redirected by the WIC and sent on the interconnecting backplane to TPCs. In this way, control actions such as fault application or setpoint adjustment can be made dynamically on the workstation. The WIC does not actively participate in the power system solution but instead functions as an interface and simulation control device [20].

2.2.3 INTER-RACK COMMUNICATION CARD

The third and final type of card in the RTDS architecture is the IRC. In multi-rack simulation cases, the data exchange between processors residing on separate racks is accomplished through special high-speed communication channel mounts on the IRCs. Each IRC includes up to four separate transmitter/receiver channels hence allowing direct connection to as many as four other racks. RTDS simulators comprised of more than five racks are therefore not fully connected if only one IRC is used per rack [20].

2.3 RTDS SOFTWARE

In addition to the specialized hardware described in the preceding section, much effort was invested in the development of appropriate software. In general terms, RTDS software falls into one of three distinct categories or levels:

- Level 1 - graphical user interface (GUI)

- Level 2 - compiler/operating system

- Level 3 - power system components/solution

2.3.1 GRAPHICAL USER INTERFACE (GUI)

All interaction between the user and the RTDS simulator is performed using a sophisticated, graphically-driven user interface program (PSCADTM[17]). PSCAD is a family of software tools consisting of individual modules used to accomplish various tasks in the overall operation of the RTDS. Fig. 2.2 provides an overview of PSCAD and how its modules are related.

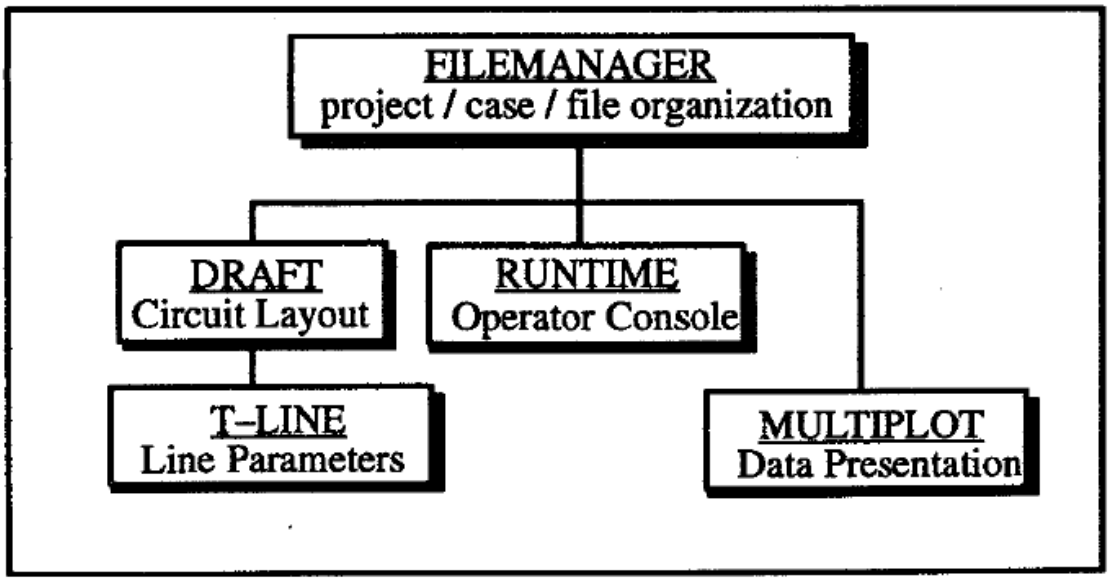


Figure 2.2: PSCAD Software Modules[20]

The FILEMANAGER module represents the top level of PSCAD through which the user enters the graphically driven system. This icon-based file management system aids the user in maintaining a large number of studies and the many files associated with them. All other PSCAD modules are invoked from within the FILEMANAGER level. DRAFT is a pre-processing module used to assemble the power system circuit and enter its associated parameters. Instead of creating a text-based file describing the interconnection of the network under study, the user simply draws a picture of the circuit. Icons representing individual power system components are arranged on one side of the workstation screen in the form of a user-defined library and the circuit is assembled on the other. Assembly is accomplished by choosing and copying library components, dragging them to the circuit assembly area, and interconnecting them in an appropriate manner. All actions are mouse driven and can be performed with minimal effort and maximum speed. Once circuit assembly and data entry is complete, the user can save and compile the circuit for simulation.

A special software module exists to facilitate entry of transmission line data. This so-called T-Line module accepts as input information relating to line geometry and conductor type. Operation of the RTDS is accomplished using the RUNTIME module. One or more operator's consoles can be created through which individual simulation cases can be downloaded, started and controlled. During the simulation the user can monitor specified system quantities using graphical icons of meters and plots. In addition, it is also possible to dynamically interact with the simulation as it runs by creating push buttons, set point sliders, switches etc.

As an example, a fault can be applied at a pre-defined point in the power system using a push button. The type, location and duration of the fault are defined in DRAFT before compiling the case. In addition, other events associated with the fault application (e.g. timed breaker operation) are also defined prior to compiling through the DRAFT sequencer. A detailed examination of fast changing or transient signals is accomplished using the plot facilities resident in RUNTIME. Plots are updated only at the request of the operator or upon the occurrence of an operator initiated transient (eg. application of a fault from RUNTIME). The signals displayed on a plot are captured in real time on the RTDS WIC and stored in its local memory until the specified capture interval has elapsed. Once the entire event has concluded, the stored data is transferred from the

WIC to the appropriate plot component on the workstation screen and displayed. Data captured and displayed using the RUNTIME plot icon can also be saved and stored on the workstation for further analysis. MULTIPLOT, the final module of PSCAD is used to analyze and evaluate captured waveforms as well as prepare output plots in a report-ready format. Several useful functions available in MULTIPLOT include Fourier analysis and total harmonic distortion computation.

2.3.2 RTDS COMPILER

An important link between the graphical user interface software and the DSP code which runs on the RTDS is a specially designed compiling system. The RTDS compiler takes the circuit layout and parameters entered through DRAFT and produce all of the parallel processing code required by the digital signal processors. In addition, the compiler automatically assigns the role that each DSP will play during the simulation based on the required circuit layout and the available RTDS hardware. Although the compiler follows a number of general rules when allocating processors, it is not always obvious to the user which processor is performing which tasks. Since under many circumstances interconnections to analogue and digital I/O are required, the user must know where to access the appropriate signals. For this reason, in addition to producing the DSP code, the compiler also produces outer-readable file indicating the function of each processor in the particular case being considered. This so-called MAP File also directs the user to I/O points which might be required for interfacing of physical measurement, protection or control equipment. In the case of multi-rack simulation cases, allocation of available RTDS hardware must follow a set of specific rules. One of the most important aspects in allocation of racks is the recognition of tightly coupled subsystems within the overall power system. Since the solution algorithm applied in the RTDS is based on nodal admittance matrix techniques, each electrical node introduces one row and one column in the overall system matrix. As the number of nodes increases, so does the size of the matrix and the computations required. Recognizing that travelling wave transmission line models break up the nodal admittance matrix into block diagonal portions, the number of computations can be reduced. In fact, the block

diagonal portions of the overall matrix can be solved independently, hence allowing equations for different portions to be solved in parallel. This mathematical decoupling lends itself well to parallel processing techniques and is thus exploited in the RTDS design. Subsystems of tightly coupled components can be identified and assigned to different RTDS racks in order to reduce the computational burden on processors [20].

2.3.3 POWER SYSTEM COMPONENT MODELS

In order to produce the DSP code, the compiler accesses a library of predefined and pre-assembled power system component models. Since these software models will directly impact the minimum simulation timesteps achievable, they are written in low level machine language code. All of the code used in conjunction with the component models is hand as sembled and optimized. Although this approach is somewhat labour intensive, it need only be doneonce for each component. The optimized models are then stored in the library for access by the compiler. The following power system component models currently exist with the RTDS [20].

- passive R, L, C components

- single and twin circuit transmission lines

- 2 and 3 winding transformers c/w saturation and hysteresis

- circuit breakers and faults

- voltage sources equivalent impedance (ac, dc, harmonic)

- synchronous machines c/w exciter, stabilizer, governor,

- MOV-protected series capacitors for lines

- Static VAr compensators clw generic controls

- measurement transducers including CT and CVT models.

2.4 RTDS APPLICATIONS

Since the RTDS combines the real-time operating properties of analogue simulators with the flexibility and accuracy of digital simulation programs, there are many areas where this technology has been successfully applied. In addition, because the RTDS is housed in one or more standard 19 inch cubicles, it can conveniently be taken to substation where equipment can be tested. This feature is unique to compact digital simulation systems which operate in real-time.

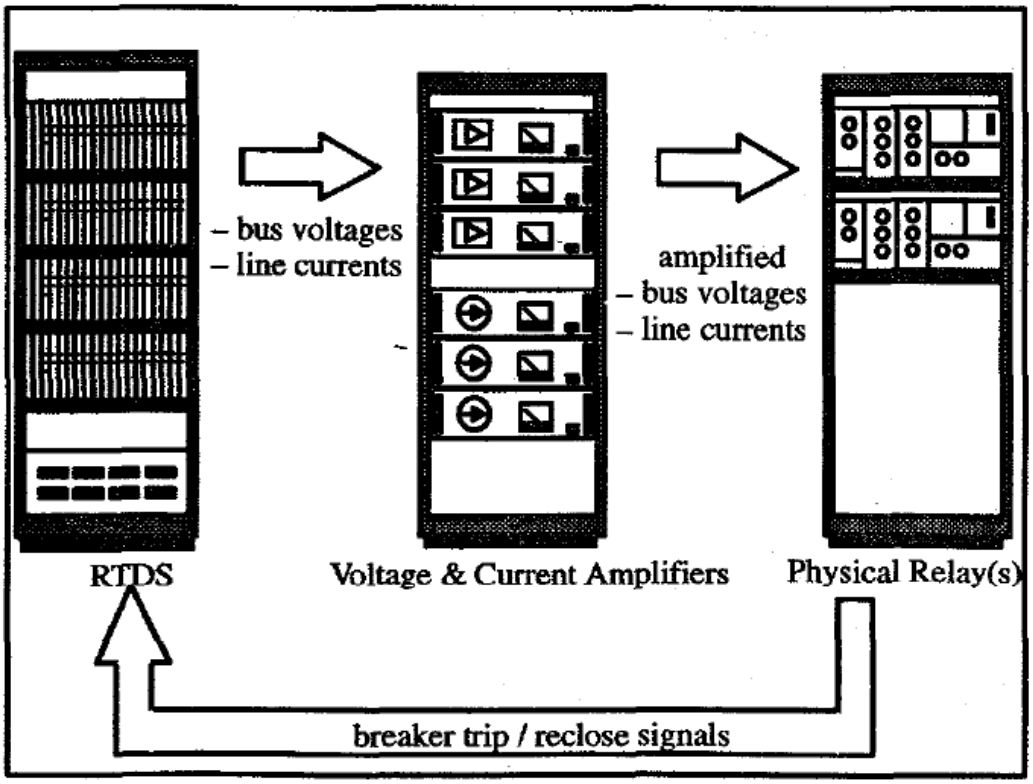


Figure 2.3: Relay Test Facility [20]

2.4.1 PROTECTIVE RELAY TESTING

When the RTDS is combined with suitable voltage and current amplification systems it can be used to perform closed-loop relay tests. A wide variety of tests can be performed, ranging from application of simple voltage and current waveforms through to complicate sequencing within a detailed power system model. Because an extensive power system component library, including measurement transducers, is available on

the RTDS, the relay can be tested under system conditions similar to those it will encounter in the real power system. During all tests, the relay can be connected via analogue output channels to voltage and current amplifiers. Relay output contacts can in turn connected back to circuit breaker models using the RTDS digital input ports. Fig. 2.3 illustrates the general concept of closed-loop relay testing using the RTDS. Closed-loop testing is unique to real-time simulation systems in that it provides a method of evaluating not only the performance of the relay, but also the response of the system to its reaction. Because of the increasing complexity of today's modem power systems, it is becoming increasingly important to study effects and interactions of various system components on one another. Closed-loop equipment testing using real-time digital simulators is a good example of how this can be accomplished in a relatively straight-forward manner.

2.4.2 CONTROL SYSTEM TESTING

In a manner similar to that described for protection system testing, the RTDS can be applied to the evaluation and testing of physical control equipment. Required analogue and digital signals produced during simulation of the power system can be connected directly to the control equipment. The controller outputs are then connected to input points on the particular power system component model being simulated. This again closes the test loop and allows evaluation of control equipment performance on the system to which the equipment is tested. Examples of typical applications include detailed testing of HVDC control equipment, static VAr compensators (SVC) controls as well as controls associated with synchronous machines. Fig. 2.4 illustrates a typical configuration for HVDC control system tests where analogue voltage and current signals are passed to the control equipment which in turn issues firing pulses to HVDC converter valves being represented within the power system model.

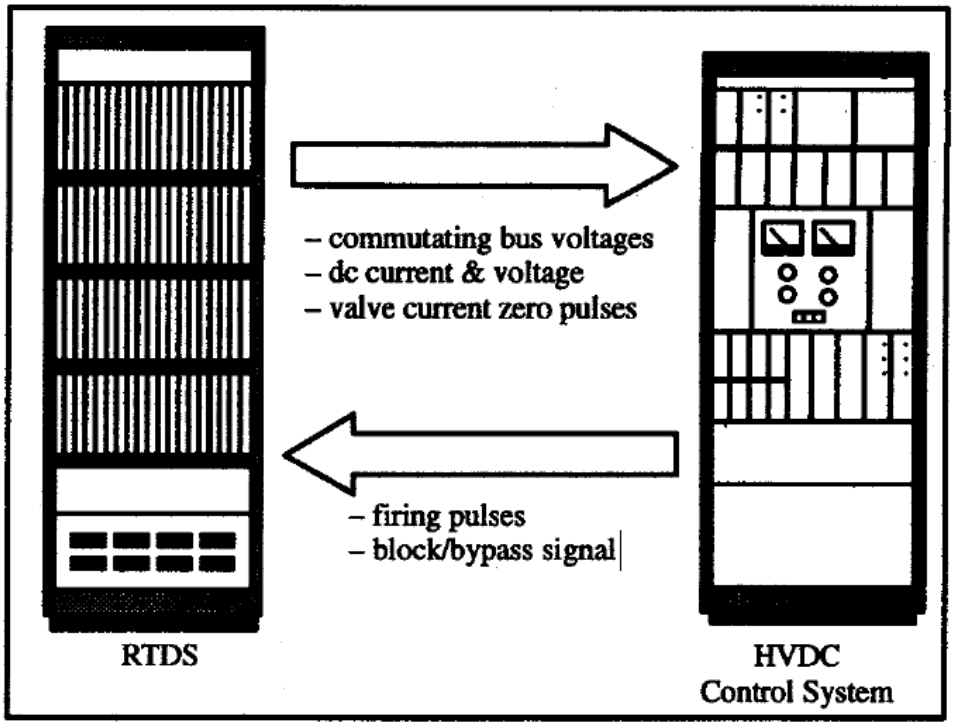


Figure 2.4: HVDC Control System Testing [20]

2.4.2.1 ANALOGUE SIMULATOR EXPANSION

It has been shown [23] [24] that the RTDS can be used to expand and enhance the capabilities of traditional analogue simulators if appropriate interfacing points and methods are chosen. Recently, particular attention has been paid to expansion of HVDC analogue simulators to permit representation of large ac networks at either end of the HVDC link. Due to the inherent difficulties associated with ac system modeling in analogue simulators, interfacing the RTDS to an analogue simulator offers a simple and relatively inexpensive alternative.

2.4.2.2 GENERAL APPLICATION

Since the RTDS includes a comprehensive library of ac and dc power system component models and since it can be continuously enhanced by adding new software modules, it provides a flexible method for studying all aspects of power system development, design and operation. Furthermore, since the **RTDS** responds in real-time to events initiated through the user interface software (i.e. operator's console), it provides an excellent method for training operators and educating engineers in the principles of power system operation [20].

2.5 HARDWARE-IN-THE-LOOP SIMULATION

Main features of RTDS are hardware-in-the-loop (HIL) tests and simulations. The benefits of subjecting equipment to simulated environments include increased flexibility in designing, implementing, and executing experiments that reflect situations as close as possible to real world conditions. Individual test scenarios can be quickly configured and adapted once the device under test is connected, as only changes in software models need to be accommodated. During early design stages, offline simulation softwares are used for verification and performance evaluation of the design concepts. Thereafter, the implementation stage begins where the control algorithm is realized on a physical platform. A hardware-in-the-loop (HIL) arrangement of a physical controller platform, in association with a real-time simulator, is needed to safely and thoroughly verify the design integrity and evaluate its functions and performance. This type of HIL testing, involving the exchange of waveforms between the simulator and the controller platform at the signal level with no real power exchange, is called controller hardware-in-the-loop (CHIL) testing. Consequently, the required time and financial expenses associated with experimentation and testing are reduced. Further, the risk of late and costly changes due to specifications and design choices can be avoided as thorough and realistic tests can be performed earlier in the development cycle and before actual installation and deployment in the field. And, HIL testing reveals how equipment will interact with the system for which it is being targeted in a way that more conventional non-interactive testing approaches cannot. The complexity of the so called plant under control is included in test and development by adding a mathematical representation of all related dynamic systems [26] [27]. The existing real-time simulators utilize parallel processing, where multiprocessors, either General Purpose Processors (GPP) or Digital Signal Processors (DSP), or computer clusters are utilized, to achieve the required real-time simulation performance. The system to be simulated is divided into zones, and the mathematical model of each zone is assigned to a given processor. A master processor controls the communication and maintains synchronism among the other processors. In such paradigm, the minimum simulation timestep is dictated by the speed and number of processors, the synchronization and communication overhead time among the processors, and the efficiency of the software code. However, existing real-time simulators at CAPS has limitations on the minimum timestep size which is roughly

between 2 and 50 microseconds, the frequency bandwidth of the simulation results which is approximately up to 10 kHz [37].

An HIL simulation must include electrical emulation of sensors and actuators. These electrical emulations act as the interface between the plant simulation and the embedded system under test. The value of each electrically emulated sensor is controlled by the plant simulation and is read by the embedded system under test (feedback). Likewise, the embedded system under test implements its control algorithms by outputting actuator control signals. Changes in the control signals result in changes to variable values in the plant simulation. An example is the HIL simulation of power system relays, sometimes referred to as controller hardware-in-the-loop (CHIL) simulation. Multiple issues must be addressed for the fidelity of such a simulation. The simulation time step must be chosen to account for the dynamics of the studied system. Measurement transducers must have sufficient bandwidth and digital and analog interfaces must not introduce excessive latency into the simulation. The introduction of a high power physical device into the real-time simulation is referred to as power hardware-in-the loop (PHIL) and is now evolving into the megawatt range. In this technique, amplifiers are present to effect a large change in the physical environment from a signal-level change in simulation [25]. The PHIL and CHIL concepts are shown in Fig. 1.3.

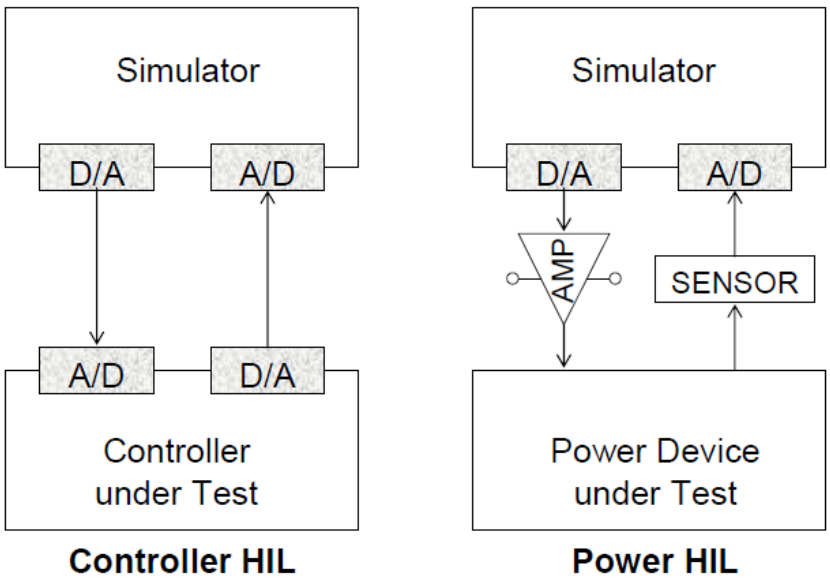


Figure 2.5: Structural distinction between CHIL and PHIL simulations [25]

For example, a HIL simulation platform for the development of automotive anti-lock braking systems may have mathematical representations for each of the following subsystems in the plant simulation:[1]

- Vehicle dynamics, such as suspension, wheels, tires, roll, pitch and yaw;
- Dynamics of the brake system's hydraulic components;
- Road characteristics.

In many cases, the most effective way to develop an embedded system is to connect the embedded system to the real plant. In other cases, HIL simulation is more efficient. The cost of the approach should be a measure of the cost of all tools and effort. The duration of development and testing affects the time-to-market for a planned product. Safety factor and development duration are typically equated to a cost measure. Specific conditions that warrant the use of HIL simulation include the following:

- Enhancing the quality of testing
- Tight development schedules
- High-burden-rate plant
- Early process human factor development
- Enhancing the quality of testing

Usage of HILs enhances the quality of the testing by increasing the scope of the testing. Ideally, an embedded system would be tested against the real plant, but most of the time the real plant itself imposes limitations in terms of the scope of the testing. For example, testing an engine control unit as a real plant can cause dangerous conditions for the test engineer. HILs provides the efficient control and safe environment where test or application engineer can focus on the functionality of the controller [27].

Hardware-in-the-Loop Simulation for Power Electronics systems is the next quantum leap in the evolution of HIL technologies. The ability to design and automatically test power electronics systems with HIL simulations will reduce development cycle, increase efficiency and improve reliability and safety of these systems for large number of applications. Indeed, power electronics is an enabling technology for hybrid electric

vehicles, electric vehicles, variable speed wind turbines, solar photovoltaic, industry automation, electric trains etc. [38].

The Real Time Digital Simulator, or RTDS, provides power systems simulation technology for fast, reliable, accurate and cost-effective study of power systems with complex High Voltage Alternating Current (HVAC) and High Voltage Direct Current (HVDC) networks. The RTDS Simulator is a fully digital electromagnetic transient power system simulator that operates in real time. Power electronics systems are a class of dynamic systems that exhibit extremely fast dynamics due to high-frequency switching action of power electronics switches (e.g. IGBTs, MOSFETs, IGCTs, diodes etc.). Real-time simulations of switching transitions require digital processor speeds and latencies that can actually be met with off-the-shelf computer systems and with FPGA/CPU platform technologies making it 100 times faster than traditional computational methods to achieve high-resolution HIL for power electronics.

Because the simulator functions in real time, the power system algorithms are calculated quickly enough to continuously produce output conditions that realistically represent conditions in a real network. Real-time simulation is significant for two reasons -- the user can test physical devices and the user is more productive by completing many studies quickly with real-time simulation.

Additionally, RTDS can be connected directly to power system control and protection equipment. For example, it can be used to test HVDC (High Voltage Direct Current) controllers or protective relays. Testing on an RTDS Simulator is more thorough than other test methods because the user is able to subject the equipment to many severe but realistic conditions that could not possibly be achieved when it is installed on the physical system [38].

3 HARDWARE

This chapter gives a short overview over the used Hardware and the requirements. The criteria for the experimental setup and why it was chosen are also explained.

3.1 BACKGROUND

The Real Time Digital Simulator (RTDS) typically communicates signals from a real time simulation to externally connected hardware components via fiber-optic media. To communicate analog signals, fiber-optic cable is run to a digital-to-analog converter, which decodes the fiber-optic protocol and translates it into an analog voltage signal. Similarly, analog signals can also be sent into the RTDS simulation.

The RTDS (Real Time Digital Simulator) is a special purpose multi-processor computer system optimized for power system simulations. It is designed for real-time simulation, which means that the time-steps of the simulated system advance at the same rate as standard time in the real world, thus giving the simulator the ability to run "real-time" simulations. The real-time processing and the large number of input-output channels facilitate hardware-in-the-loop simulations, wherein a software simulation interacts in real time with physical equipment, such as the dynamometers or the 5 MW power electronic based Variable Voltage Source (VVS) in the CAPS facility. As an example, a superconducting cable, power electronics converter, or motor can be loaded and its thermal response investigated using real time loading profiles, as would occur in an electric utility industry or in shipboard power and propulsion systems. In return, the equipment connected in this manner interact to the "virtual" system in a realistic way.

The RTDS typically connects to external equipment via analog to digital and digital to analog connectors. It is desirable to improve on analog measurements and control signals. To expand the possibilities of the RTDS a project was set up to create a digital way of communication with a standard based protocol which offers multi-channel possibilities, low-latency and high-throughput. This should allow higher flexibility for test purposes and better opportunities for test benches.

Often, many signals in the RTDS simulation must be exchanged with components outside the RTDS simulation. For instance, multiple control software implementations

running outside the RTDS may require such signals to be sent/received from power electronics being simulated within the RTDS. In such a case, individual wires can be used to interface the RTDS and the control software implementations. While a connecting a few wires may be feasible, as the number of wires (signals) grows, the solution of running individual wires for each signal becomes prohibitively inefficient.

In order to make it easier to facilitate many-signal digital communication between internal simulations and external hardware-in-the-loop components, RTDS provides a Xilinx FPGA board capable of decoding/encoding the RTDS fiber-optic protocol. This Xilinx board, with the associated programming to communicate signals between an RTDS simulation, is known as a GTFPGA. The GTFPGA provides an FPGA implementation that allows 128 four-byte signals to be sent and received (64 four-byte signals in each direction) between a running RTDS simulation.

3.2 HARDWARE REQUIREMENTS

After a test case was defined, which included Herb´s controller board that will be described in more detail later, the project was set up to enable a pure digital communication of transmitting and receiving signals from the RTDS. Herb´s controller board is a flexible prototyping control platform being developed for grid connected power electronic converters. It communicates to upper logic through a high speed serial link based on the Rocket I/O technology from Xilinx and uses the Aurora 8B/10B communication protocol. For this purpose it was necessary to have an FPGA board capable of data transfer over two optic fiber channels at the same time; to the RTDS and Herb´s controller board.

CAPS had already been working with the smaller ML 507 FPGA board from Xilinx, thus the decision was made to stay with Xilinx and look for a development board with higher flexibility, namely, one with more than one fiber port. Based on this criterion, the ML 605 Xilinx board was selected, as it not only has one fiber port already on board, but also the expansion capability of another 8 fiber ports with an additional extension board (FM-S18).

3.2.1 ML 605 XILINX

The Virtex®-6 FPGA ML605 Evaluation Kit (see Fig. 3.1) is the base Xilinx platform for developing system designs demanding high-performance, serial connectivity, and advanced memory interfacing. This yields design applications for markets such as wired telecommunications, wireless infrastructure, broadcast, and many others. Integrated tools help streamline the creation of elegant solutions to complex design requirements.

The ML605 Evaluation Kit is based on the XC6VLX240T-1FFG1156 Virtex-6 FPGA, this FPGA contains 241,152 logic cells, a rating that reflects the increased logic capacity offered by the 6-input LUT architecture [28].

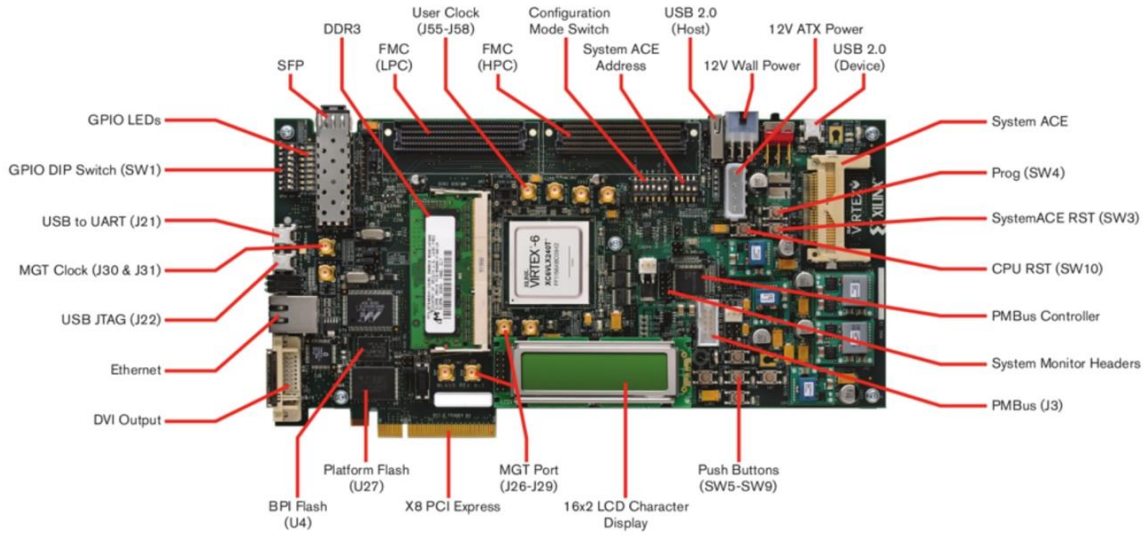


Figure 3.1: Schematic Xilinx ML 605 [28]

3.2.2 EXTENSION BOARD FM-S18

The FM-S18 is an extension board for the ML 605 and expands the given base platform's functionality by adding 8 more fiber ports and two dedicated clock generator for different connection purposes. Both clock generates can be set onboard with a Dip-Switch and are separated from the main FPGA board. The FM-S18 is connected through a FMC (HPC) port (see Fig. 3.2) and adjusted. This extra board was needed to offer the ability to connect the fiber channel of RTDS and Herb´s controller board (following chapter) at once. [29]

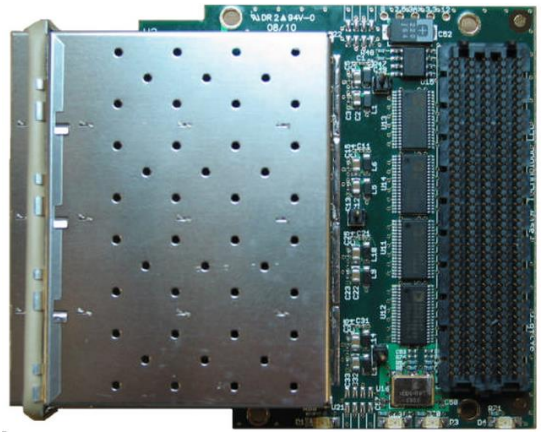


Figure 3.2: Picture of Xilinx Fm-S18 [29]

3.2.3 HERB GINN´S CONTROLLER BOARD (HCB)

This flexible prototyping control platform is being developed for use in grid connected power electronic converters that will be very useful for R&D groups at universities, research labs, and industries etc. The new modularized control platform is based on an old controller board developed previously by Dr. Herb Ginn (USC, Columbia) that had limited flexibility and applications. Based on the name of its developer, the flexible prototyping control platform came to be known as "Herb´s Controller Board" (see Fig. 3.3) around CAPS, and for simplicity's sake, this is what it will be referred to in this document.

The new control platform has 3 system modules: **Controller Board**, **Hardware Manager Board** and **Adapter Board**.

The Controller Board and Hardware Manager Boards have some common functionalities as well as independent functions. They are connected over a gigabit serial backplane using optical channels. This allows for various system configurations, making system architecture very flexible.

Because of technological advances, high speed serial communication channels have better features than traditional bulky, expensive, noise prone parallel communication channels. Serial channels which provide data rates over one gigabits/sec allow for high speed serial backplanes in the system with great flexibility.

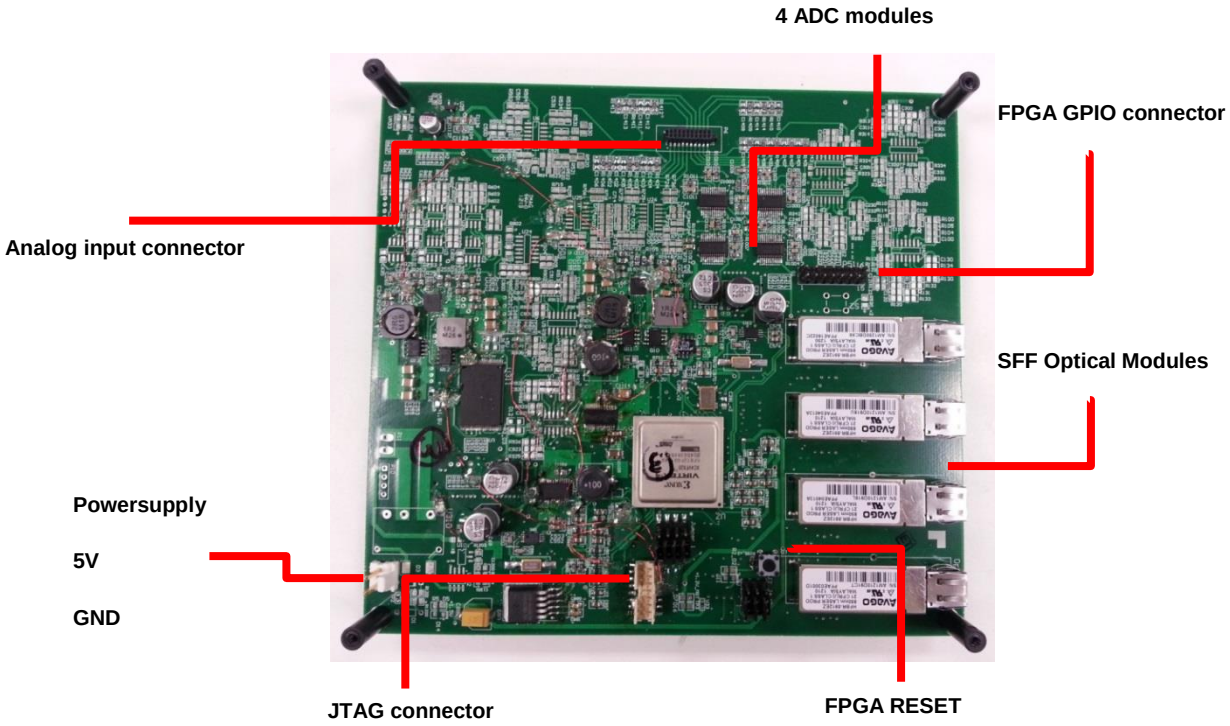


Figure 3.3: Herb´s controller board

The use of standard control modules serves to allow researchers to focus only on control subsystems of interest, thus the University of South Carolina developed a structured modular system approach optimized for use in large complex systems. Modules are reusable and reconfigurable for future projects; they can be represented graphically and interconnected by signal flow lines. The use of existing modules has the benefit of providing a known starting point for design which can then be modified or swapped out to meet desired system requirements.

Modulation is performed by the DSP and gate pulses are transferred from the controller to the FPGA, which requires a very high precision control of PWM signals. A High Speed serial channel allows for precise control of PWM signal placement to the gate driver circuit, minimizing distortions caused by jitter. To support the high speed serial data communication and to reduce EMI and noise problems, fiber optic modules are used on Controller and Hardware manager boards. The hardware manager board has the Xilinx FPGA as a core device, which includes inbuilt high speed transceivers called Rocket I/O. Xilinx also has IPcore solutions for high speed serial protocols such as Aurora protocol. This Aurora protocol along with features of Rocket I/O provides a highly reliable and efficient communication channel.

4 SOFTWARE

This chapter describes the main *inputs* and *outputs* of each software block and their function in the whole program. The program contains 6 software blocks which were adapted from example programs, modified from other projects or build from scratch, and then interfaced to each other and the real physical pins by signals through a header file.

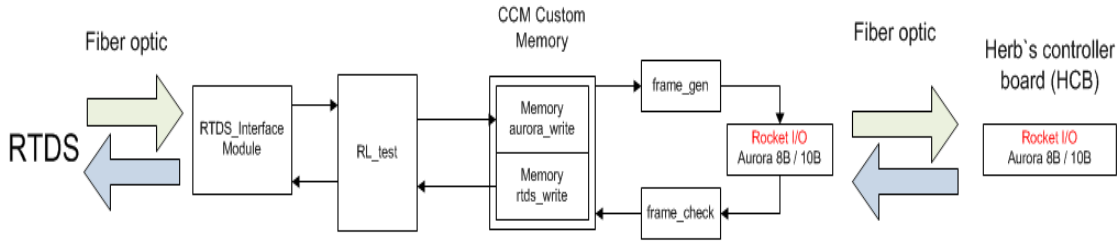


Figure 4.1: Data flow diagram of the program

The main program's data flow is described in Fig. 4.1. Data from the RTDS gets transferred via optic fiber link to the Xilinx ML 605 FPGA board where the RTDS_InterfaceModule converts the high speed serial bit stream into 32 bit integer values, each structured with a 4 bit address up front. These 32 bit integer values get read by the RL_test block and stored into the rtds_write array in the CUST_MEM block. The Custom memory block functions as a buffer for the two different clock speeds (125MHz on RTDS side and rtds_write array ←→ 250 MHz on Rocket I/O side and aurora_write array) of the program and guarantees the integrity of consecutive data.

The frame_gen block reads the data from the rtds_write array and splits it into a data stream of 8Bit values, which are forwarded to the Rocket I/O block. There, the data gets masked with the Aurora 8B/10B protocol and sent via optic fiber cable to Herb's Controller board.

Herb`s controller board unmasks the Aurora 8B/10B data and recombines it to form the original 32 bit values sent from the RTDS.

The data transfer from Herb's board to the RTDS perdorms in a similar fashion. Data gets captured by the Rocket I/O core on Herb's controller board and masked to the Aurora 8B/10B standard. The data is then transferred via optic fiber to the Rocket I/O core on the ML 605. After unmasking the data back to 8 bit values, the frame_check block accomplishes the recombination back to 32 bit values and stores them into the aurora_write array in the CUST_MEM block. The data is then read by the RL_test block and transferred to the RTDS_InterfaceModule structured with an address up front to make sure the RTDS receives the right data at the right port.

4.1 RTDS INTERFACEMODULE

The RTDS_InterfaceModule is a proprietary block developed by ABB and it cannot be accessed directly. This block protects the operations and knowledge of the real RTDS fiber optic transfer protocol and is controlled by input and output signals to form the RL_test block. Its main process is to transform a bit stream from the RTDS into a stream of 16 bit addresses and 32 bit values that can be understood by the RL_test block (see Fig. 4.2).

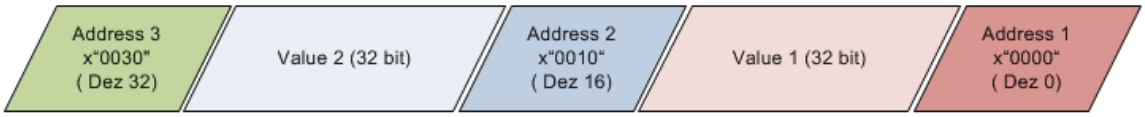


Figure 4.2: Schematic of the data configuration after the RTDS_InterfaceModule

4.2 RL_TEST

The RL_test block was taken and modified from a project which worked with data transfer from the RTDS over Rocket I/O and the PCIe interface. Modifications were made in such a way that data output (see fig. 4.3) has the right format to be stored in the Custom Memory. The main in- and outputs and their purpose will be described below.

```
entity RL_test is port(
    -- to MGT user interface
    UserTxAdr           : out std_logic_vector(7 downto 0);
    UserTxData          : out std_logic_vector(31 downto 0);
    UserTxWr            : out std_logic;
    UserTxFull          : in std_logic;
    UserTxInProgress    : in std_logic;
    UserLockBank        : out std_logic;
    UserFreeBank        : out std_logic;
    UserVersion         : out std_logic_vector(7 downto 0);
    -- from MGT user interface
    UserRxAdr           : in std_logic_vector(15 downto 0);
    UserRxData          : in std_logic_vector(31 downto 0);
    UserRxValid         : in std_logic;
    UserTstepPulse      : in std_logic;
    -- misc
    CaseReset           : in std_logic;
    Clk100M             : in std_logic;
    Rst                 : in STD_LOGIC;
    -- to/from CUST_MEM
    RI_ADDR_R : out std_logic_vector(7 downto 0);
    RI_ADDR_W : out std_logic_vector(7 downto 0);
    RI_DI : in std_logic_vector(31 downto 0);
    RI_DO : out std_logic_vector(31 downto 0);
    RI_RE : out std_logic;
    RI_WE : out std_logic;
```

Figure 4.3: Rl-test program code, signal ports

Inputs to receive data from the RTDS

UserRxValid: A value which is set by the RTDS_InterfaceModule and works as a boolen indicator of the veracity of the incoming data.

UserRxAdr: The address provided by the RTDS_InterfaceModule for each data package. The RTDS starts at x”0000” and sequentially adds dez”16” (x”0010”) each time for another 32 bit value. Because of the limitation of the RTDS which can only send 64 32bit values at each time step, the bit length is shortened to 8bit.

UserRxData: Corresponding 32 bit values get stored in this variable each time UserRxValid is 1 (true).

RI_DO: UserRxData, outgoing data to the Custom memory block.

RI_ADDR_W: Address in the Custom memory where the data is stored. During the test phase, it uses the same address as UserRxAdr.

RI_WE: Write enable, goes high (1, true) when RI_DO and RI_ADDR_W are ready to be stored in the Custom memory block (rtds_write array)

Outputs to transfer data to the RTDS:

UserTxAdr: The same logic as UserRxAdr is used to mask the 32 bit values with the correct addresses up front to display at the first, second, third, etc. in the RTDS main program. The generated addresses start at x”0000” and are raised by dez”16” (x”0010”) each time for another 32 bit value.

UserTxData: The correct 32 bit value corresponding to the address is stored in this variable.

UserTxWr: This variable is normally 0. Once the correct address and data are available in UserTxAdr, UserTxData goes high (1, true) so the whole package is transferred at once to the RTDS_InterfaceModule.

RI_DI: Input data from the custom memory block (aurora_write) which is forwarded to the UserTxData variable.

RI_ADDR_R: Address in the Custom memory block where the data is read. During the test phase, it occupies the same address as UserTxAdr.

RI_RE: Read enable, goes high (1, true) when RI_DI and RI_ADDR_R are ready to be read from the Custom memory block (aurora_write array)

4.2.1 INCOMING DATA FROM RTDS

```
--------------------------------------------------------------------------
---- Decode incoming Data from RTDS
----    - iToGT1 (UserRxAdr = x"0000"), iToGT12(UserRxAdr = x"0010"), ...
----
---- Write incoming data from RTDS to AURORA memory.
--------------------------------------------------------------------------
process(Clk100M,Rst)
    constant nr_To_vals : natural := 64;
begin

    if Rst = '1' then   .
        RI_WE <= '0';
        GPIO_LED        <= '1';
    elsif rising_edge(Clk100M) then
        RI_WE <= '0';

        if UserRxValid='1' then
            -- UserRxAdr is always incremented by 16
            for i in 0 to (nr_To_vals - 1) loop
                -- UserRxAdr is byte addressed
                if (UserRxAdr = (i*4)) then
                    RI_DO <= UserRxData;
                    RI_ADDR_W <= UserRxAdr(7 downto 0);
                    RI_WE <= '1';
                end if;
            end loop;
        end if;
    end if;
end process;
```

Figure 4.4: RL-test program code, receiving from RTDS logic

nr_To_vals displays the maximum number of values which can be read from the RTDS with this method (see Fig. 4.4). If the UserRxValid goes high, the data is available, the correct address is calculated, and the incoming 32 bit value is transferred from the variable UserRxData to RI_DO. The address is also transfered to RI_ADDR_W. After both variables are set, RI_WE goes high so the data can be stored at the corrct location in custom memory.

4.2.2 SEND DATA TO RTDS

```
---- Read from AURORA memory and send Data to RTDS
----

process(Clk100M,Rst)
    variable lAddr: std_logic_vector(7 downto 0);
begin
    if Rst='1' then
        UserTxAdr <= (others => '0');
        UserTxData <= (others => '0');
        UserTxWr <= '0';
        SendToRTDS_state <= 0;
        UserLockBank <= '0';
        UserFreeBank <= '0';

        RI_RE <= '0';
    elsif rising_edge(Clk100M) then
        UserLockBank <= '0';
        UserFreeBank <= '0';
        UserTxWr <= '0';
        RI_RE <= '0';

        case SendToRTDS_state is
            when 0 => -- IDLE
                -- if start of time step
                if UserTstepPulse = '1' then
                    SendToRTDS_state <= SendToRTDS_state + 1;
                end if;

                RI_ADDR_R <= std_logic_vector(to_unsigned(SendToRTDS_state*4, RI_ADDR_R'length));
                RI_RE <= '1';

            when 1 => -- LOCK_BANK
                UserTxWr <= '1';
                UserLockBank <= '1';

                RI_ADDR_R <= std_logic_vector(to_unsigned(SendToRTDS_state*4, RI_ADDR_R'length));
                RI_RE <= '1';

                SendToRTDS_state <= SendToRTDS_state + 1;
```

Figure 4.5: Rl-test program code, sending to RTDS logic

The RTDS bank gets locked, causing a RTDS data stream to be generated. SendToRTDS_state represents a value of how many 32 bit variables are transferred from the Xilinx ML 605 to the RTDS (see Fig 4.5). After the whole data stream is generated with the corresponding addresses and 32 bit values, the RTDS bank is freed and the whole bit stream is transferred to the RTDS_InterfaceModule to be sent to the RTDS.

4.3 CUST_MEM

The custom memory block consists of two arrays, rtds-write and aurora_write. Each array is 64 indexes big, each index is a 32 bit value. The aurora_write array is clocked with the Rocket I/O clock with 250 MHz and the rtds_write array is clocked with the RTDS clock (CLK100M) with 100 MHz. Each array gets triggered for read and writes commands from the associated data transfer block. The RL-test block controls the transmitting and receiving signals and commands for data exchange with RTDS and transmitting and receiving on the Rocket I/O side it's either the frame_gen or frame_check block (see Fig. 4.6).

```
entity CUST_MEM is port (
    AURORA_CLK : in std_logic; -- ROCKET I/O clock
    AURORA_ADDR_R : in std_logic_vector(7 downto 0); -- reading from rtds_write
    AURORA_ADDR_W : in std_logic_vector(7 downto 0); -- writing to aurora_write
    AURORA_DI : in std_logic_vector(31 downto 0); -- DATA input to memory
    AURORA_DO : out std_logic_vector(31 downto 0); -- DATA output from memory
    AURORA_WE : in std_logic;
    AURORA_RE : in std_logic;

    RTDS_CLK : in std_logic;  -- RTDS clock CLK100M
    RTDS_ADDR_R : in std_logic_vector(7 downto 0); -- reading from aurora_write
    RTDS_ADDR_W : in std_logic_vector(7 downto 0); -- writing to rtds_write
    RTDS_DI : in std_logic_vector(31 downto 0); -- DATA input to memory
    RTDS_DO : out std_logic_vector(31 downto 0); -- DATA output from memory
    RTDS_RE : in std_logic;
    RTDS_WE : in std_logic
```

Figure 4.6: custom memory program code, signal ports

4.4 FRAME_GEN

The frame generator block (frame_gen, see Fig 4.7) generates ,out of the 32 bit values stored in the custom memory block, a 8 bit values which are then forwarded to the Rocket I/O.

```
entity FRAME_GEN is

port
(
    -- User Interface
    TX_DATA             : out   std_logic_vector(7 downto 0);
    TX_CHARISK          : out   std_logic;
    DATA                : in   std_logic_vector(7 downto 0);
    AURORA_ADDR_R       : out std_logic_vector(7 downto 0);
    AURORA_DO           : in std_logic_vector(31 downto 0);
    AURORA_RE           : out std_logic;
    -- System Interface
    USER_CLK            : in    std_logic;
    SYSTEM_RESET        : in    std_logic
```

Figure 4.7: frame_gen program code, signal ports

TX_DATA: Output of 8 bit data segments to the Rocket I/O. Every time in a while during the whole communication time and in between the memory data segments, the Aurora serial protocol gets synchronized and TX_DATA sends a comma alignment K-character, in our case x"BC", instead of the normal data from the custom memory.

TX_CHARISK: Each time the Aurora fiber channel gets synchronized, TX_DATA sends a comma alignment K-character in our case x"bc" the TX-CHARISK port gets high (1, true) otherwise for normal data it stays low (0, false).

AURORA_ADDR_R: Address in the Custom memory block (rtds_write array) where the data is read from. Related to the agreed data protocol with Herb´s controller board, the address can configured to select the right data packages in the necessary order.

AURORA_DO: Input data from the custom memory block (rtds_write) which is forwarded to the TX_DATA variable in 8 bit segments.

AURORA_RE: Read enable, goes high (1, true) when AURORA_ADDR_R and AURORA_DO have the related values from which index data should be read from the custom memory block (rtds_write array).

4.4.1 SEND DATA TO ROCKET I/O

```
--_____________________________ SEND TO CUSTOM BOARD _____________________


process( USER_CLK )
begin
    if(USER_CLK'event and USER_CLK = '1') then
      AURORA_RE                <= '0';
        if(SYSTEM_RESET='1') then
            TX_DATA      <=  (others => '0') after DLY;
            TX_CHARISK    <= '0' after DLY;
        else
            if ((read_counter_i = 100) or (read_counter_i = 101))then
                AURORA_ADDR_R     <= x"00";
                tx_data_mem_i <= AURORA_DO;
                AURORA_RE              <= '1';

                TX_DATA      <= tx_data_mem_i(15 downto 8) after DLY;
                TX_CHARISK   <= '0' after DLY;
            else
                if((read_counter_i = 102) or (read_counter_i = 103))then
                AURORA_ADDR_R     <= x"00";
                tx_data_mem_i <= AURORA_DO;
                AURORA_RE              <= '1';

                TX_DATA      <= tx_data_mem_i(7 downto 0) after DLY;
                TX_CHARISK   <= '0' after DLY;
                else
                    if ((read_counter_i = 104) or (read_counter_i = 105))
                    AURORA_ADDR_R     <= x"10";
                    tx_data_mem_i <= AURORA_DO;
                    AURORA_RE              <= '1';

                    TX_DATA      <= tx_data_mem_i(15 downto 8) after DLY;
                    TX_CHARISK   <= '0' after DLY;
                    else
```

Figure 4.8: frame_gen program code, sending to Rocket I/O logic

The connection establishes with Herb´s controller board and the channel gets synchronized by sending as long as the read_counter_i is smaller than 100 a comma

alignment K-character, in our case x”BC”. The agreed simple communication protocol with Herb´s controller board looks like:

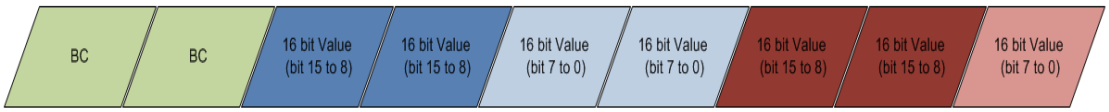


Figure 4.9: Communication protocol with Herb´s controller board

The first 8 bits of a 16 bit value gets read from the custom memory and forwarded to the Rocket I/O when the read_counter_i is 100 and 101 (see Fig. 4.8). The data gets transferred twice because of timing issues at Herb´s controller board to make sure that the right 8 bits get received. At 102 and 103 the second 8 bits of the 16 bit value gets send (see Fig. 4.9). The 8 bits can be read from the different indexes of the custom memory to make sure in the future every more complex protocol can be put together. After all 16 bit values are transmitted the block sends another 100 times a comma alignment K-character to make sure that the Aurora cores are still in sync.

4.5 ROCKET I/O (AURORA 8B/10B)

The Aurora 8B/10B protocol describes the transfer of user data across an Aurora 8B/10B channel (see Fig. 4.10). An Aurora 8B/10B channel consists of one or more Aurora 8B/10B lanes. Each Aurora 8B/10B lane is a full-duplex serial data connection. The devices that communicate across the channel are called channel partners. Figure 1-1 illustrates this relationship. The Aurora 8B/10B protocol interfaces transfer data and control to and from user applications by way of the user interface. Data flow consists of the transfer of user PDUs and user flow control messages between the user application and the Aurora 8B/10B interface, and the transfer of channel PDUs, and flow control PDUs across the Aurora 8B/10B channel [30].

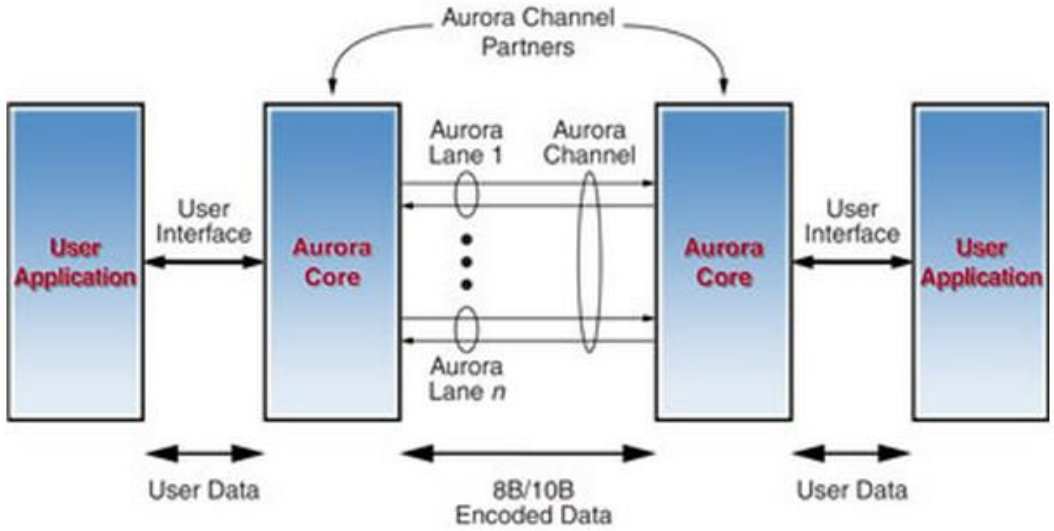


Figure 4.10: Aurora data flow chart [30]

Working example of the 8B/10B encode and decoding. For example the actual data to be sent is an 8 bit sequence which gets encoded by the Rocket I/O core (Multi-Giga bit Transceiver). The core will convert (encode) the 8-bit data into an equivalent 10-bit value based on predefined algorithm. This is done by using inbuilt algorithm/technique called 8B/10B (8 bit to 10 bit). The conversion is done to maintain DC Balance on the communication physical link. These 10 bits are then transmitted serially starting "most significant bit" (MSB) first and sent over a serial channel every clock event. At the other side the receiving MGT will check for every 10 bits. Although this

communication is synchronous (data is sent at CLK events) the CLK is not shared between TX and RX (for 2 different MGT chips) also it doesn't use any control or handshake signals. So on the receiver side there is no way to determine the 1st and 10th bit of the serial data that was transmitted. Because of minor differences in power ON times of individual TX and RX boards there could be no guarantee that 1st transmitted bit will be the 1st received bit. Here the data would probably be not lost but incorrect framing of 10-bits at the receiver would cause corrupted data. This scenario is explained in following figures [30] [31].

Example Data to be transmitted: x"0F" (hex) (see Fig. 4.11)

Equivalent 10-bit value: 010111 0100 OR 101000 1011

There are two 10-bit values available for each HEX number to maintain +Ve and –Ve DC balance of communication link. One of those two is selected depending on polarity of previous data transmitted. Polarity of 10-bit value is +Ve if number of 1's are more than 0s and –Ve if 0s are more than 1s.

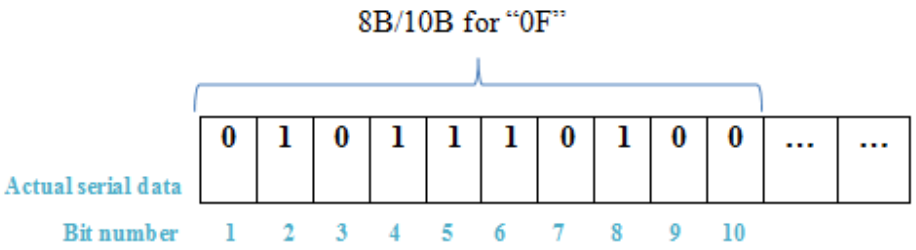


Figure 4.11: Aurora 8B/10B example data

In the same way, any HEX combination from 00 to FF has predefined +Ve and –Ve polarity 10-bit values. And one of those two will be transmitted over the serial line.

The receiver side will follow the opposite procedure. It will combine 10 incoming serial bits and convert (decode) it in equivalent 8-bit data by using 8B/10B algorithm or look up table.

To notice is that the Transmitter will start transmitting serial data as soon as it is enabled and Receiver will start receiving as soon as it is enabled. After the receiver is enabled it will capture the first 10 incoming bits and convert them into 8-bit data.

The solution to this problem is to “lock” the incoming data on receiver side to a particular frame size. Once correctly frame locked to the incoming data the communication channel will never lose synching unless the channel is disabled/ disconnected or similar issue occurred. Both cypress and Xilinx have similar 8B/10B encoder algorithms. This means for any 8 bit data both chips will create similar 10 bit data. This 10 bit conversion is needed in order to keep the DC balance of communication channels at high speed communication. If we send data without using 8B/10B encoding then for many 8 bit data combinations such as 00, 01, 10, 80, FF etc. the communication channel will have DC charge balance problems. Therefore 8B/10B encoding/decoding is necessary [30] [31] [32].

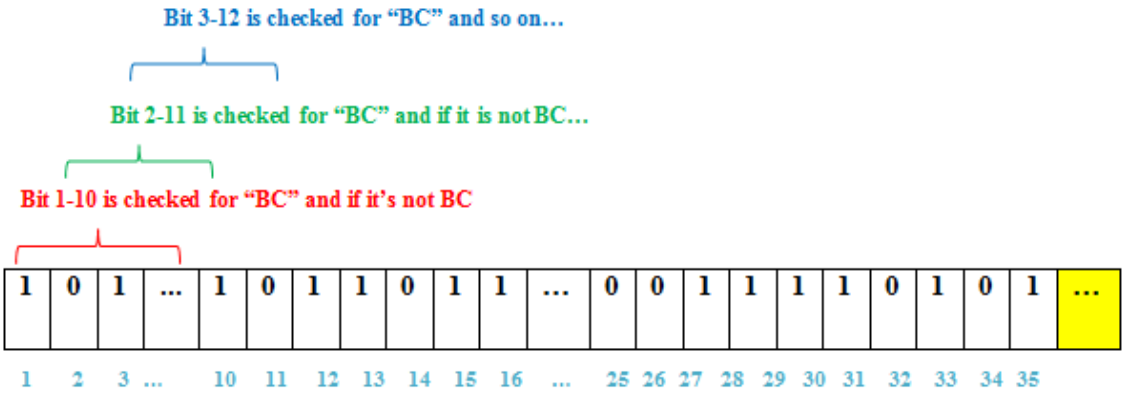


Figure 4.12: Channel alignment,”lock” channel

The receiver section will always look for consecutive 10-bits in incoming data stream. It decodes those 10 bits to 8-bits by using 8B/10B algorithm. The receiver then checks if this 8-bit data is equal to any of the “special framing characters” (most frequently used character is “BC”) defined by 8B/10B look-up table. If it is not “BC” then it will take next 10 bits and do the same operation to see if they are framing characters. The way receiver checks 10 consecutive bits looks like as followed (see Fig. 4.12). From “bit N to bit N+ 9“, in this case 1-10 then “bit N+1 to N+10” then “N+2 to N+11” and so on. If it finds “BC” for “bits N to N+9” then next 10 bits it will check are from “bit N+10 to

N+19". In this way it will keep on checking until it finds 4 consecutive "BC" characters in incoming data stream. Once it finds those 4 "BC" characters then it knows that the incoming data is frame locked and individual 10 bits in next incoming data stream are encoded 8-bit data. Here those 4 "BC" characters are needed to be sent by TX on purpose in order to frame lock the serial data [30] [31] [32].

A step by step description of how to create a certain Xilinx Rocket I/O Aurora 8B/10B core with the Xilinx core generator tool can be looked up at ANNEX D. Further specifications of the Core can be found in the relevant literature [31] [32] [33].

4.6 FRAME_CHECK

The frame_check block (see Fig. 4.13) receives the decoded 8 bit values from the Rocket I/O block and reconnects them back together to 16 or 32 bit values which are then stores in the custom memory (aurora_write array).

```
entity FRAME_CHECK is port
(
    -- User Interface
    RX_DATA                 : in  std_logic_vector(7 downto 0);
    RXCTRL_IN               : in  std_logic;
    DATA_RX                 : out std_logic_vector(7 downto 0);
    AURORA_ADDR_W           : out std_logic_vector(7 downto 0);
    AURORA_DI               : out std_logic_vector(31 downto 0);
    AURORA_WE               : out std_logic;

    RX_ENMCOMMA_ALIGN       : out std_logic;
    RX_ENPCOMMA_ALIGN       : out std_logic;
    RX_ENCHAN_SYNC          : out std_logic;

    -- System Interface
    USER_CLK                : in std_logic;
    SYSTEM_RESET            : in std_logic
```

Figure 4.13: frame_check program code, signal ports

RX_DATA: Received data from the Rocket I/O which gets stored in the custom memory block (aurora_write array) after the 8 bit sequences are put back together to a predefined value or signal.

RXCTRL_IN: Indicates if the received 8 bit value is a K-character or not. If RX_DATA is x"BC" and RXCTRL_IN is high (1, true) the core gets aligned as described in chapter 4.5 ROCKET I/O (AURORA 8B/10B).

DATA_RX: This signal is used to visualize the incoming 8 bit values to either see them as blinking GPIO LEDs on the ML 605 development board or trace them with an oscilloscope.

AURORA_ADDR_W: Address for the custom memory (aurora_write array) where the data will be stored.

AURORA_DI: Outgoing 32 bit values which are stored in the Custom memory block.

AURORA_WE: Write enable, goes high (1, true) when AURORA_ADDR_W and AURORA_DI are ready to be stored in the custom memory block (aurora_write array)

RX_ENMCOMMA_ALIGN: negative comma alignment, set high (1, true) so alignment is enabled which means Aurora 8B/10B protocol synchronizes itself, "locks" itself to an incoming data length.

RX_ENPCOMMA_ALIGN: positive comma alignment, set high (1, true) so alignement is enabled which means Aurora 8B/10B protocol synchronizes itself, "locks" itself to an incoming data length.

4.6.1 INCOMING DATA FROM ROCKET I/O

```
---- Write incoming data from ROCKET I/O to RTDS memory.
--------------------------------------------------------------------------------
process(USER_CLK,SYSTEM_RESET)
    constant nr_To_vals : natural := 12;
begin

    if SYSTEM_RESET = '1' then
        AURORA_WE <= '0';
        FrameCount_i <=x"00";

    elsif(USER_CLK'event and USER_CLK = '1') then
        AURORA_WE <= '0';

        if(rxctrl_i = '1' and rx_data_i = x"BC")then
            FrameCount_i <= x"00";
             AURORA_WE <= '0';
        else
        AURORA_WE <= '0';

        case FrameCount_i is
            when x"00" =>
                    AURORA_DI <= x"0000000" & b"000" & rx_data_i(0);
                    AURORA_ADDR_W <= FrameCount_i;
                    AURORA_WE <= '1';
                    FrameCount_i <= std_logic_vector(unsigned(FrameCount_i) + x"10");


            when x"10" =>
                    AURORA_DI <= x"0000000" & b"000" & rx_data_i(1);
                    AURORA_ADDR_W <= FrameCount_i;
                    AURORA_WE <= '1';
                    FrameCount_i <= std_logic_vector(unsigned(FrameCount_i) + x"10");
```

Figure 4.14: frame_check program code, receiving from Rocket I/O logic

As long as rxctrl_i is high (1, true) and rx_data_i is x”BC” the FrameCount_i signal stays 0 (see Fig. 4.14). If one of the two variables changes their status, a switch/case method is called which separates the different incoming bits and stores them into the custom memory. The received data fame from Herb´s controller board looks as shown below:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 1
0	0	0	0	0	0	PWM 2	PWM 1

Figure 4.15: Communication protocol from Herb´s board to the FPGA ML 605 board

An 8 bit value gets transmitted which is 0 from bit 2 to bit 7 and only the first two bits are modulated relating to two PWM signals which is generated at Herb´s controller board (see Fig. 4.15). The two PWM signals get extracted and stored at different memory indexes to be able to display them in different plots. For each time the rxctrl_i is not high, the FrameCount_i variable is added by x”10” (dez”16”), which correlates exactly with the intern RTDS communication protocol for easier data exchange.

4.7 TEST SETUP

The result and functionality of the program and setup and all the data transfomration got tested with a final loop test where two 16 bit values got generated by the RTDS (see Fig. 4.16) and send over the fiber channel to the ML 605 FPGA board. Here they got split into 8 bit sequences and sent to Herb`s controller board (HCB) with the Rocket I/O using the Arurora 8B/10B protocol. After decoding and recombining of the 8 bit values to two 16 bit values the values were taken as a reference for PWM signals and send back over the optic fiber cable to the ML 605 and from here to the RTDS. The result of the final test shows that a pure digital communication is possible and necessary for better noise reduction, better accuracy and no more grounding problems because of the electrical isolating characteristic of the optic fiber cables and has so no disadvantages in the normal test scenarios comparing to an analog way like it is now.

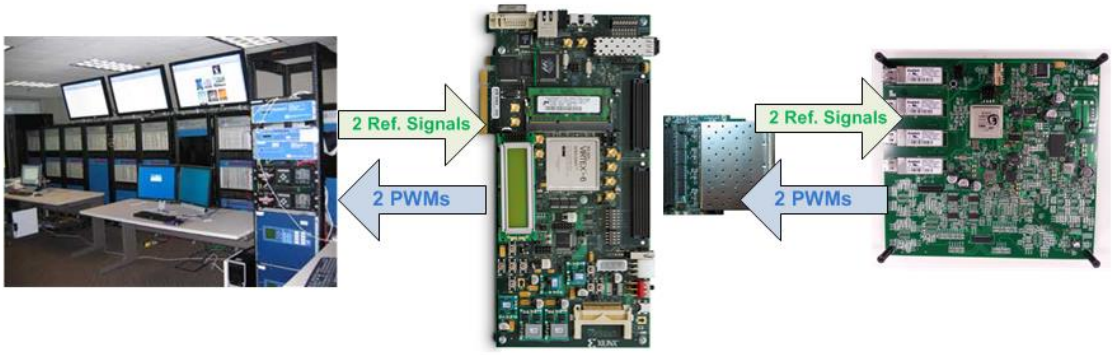


Figure 4.16: Schematic data flow chart, RTDS[3] to ML 605[4] and FM-S18[5] to HCB

[3] http://www.caps.fsu.edu/tour-RTDS.html
[4] https://www.core-vision.nl/images/Trainingen/ml605-board.jpg
[5] http://shop.trenz-electronic.de/catalog/images/products/24338klein.jpg

5 CONCLUSION

In the course of this thesis the theoretical basis for the pure digital data exchange for RTDS to external equipment was introduced. Based on these theoretical principles and the already set up test case, a program for this purpose was developed with a high speed serial communication protocol called Aurora 8B/10B. Aurora is a scalable, lightweight, link-layer protocol that is used to move data across point-to-point high speed serial links. It provides a transparent interface to the physical serial links, allowing upper layers of proprietary or industry-standard protocols to easily use these high-speed serial links. Aurora is an efficient low-latency protocol that uses the least possible amount of logic while offering a rich, highly configurable feature set. Aurora increases bandwidth and throughput through bonded lanes. The protocol encodes 8 bit data into 10 bit sequences under pre-defined algorithm as described in chapter 4.5. There are two 10-bit values available for each 8 bit value (HEX number) to maintain +Ve and –Ve DC balance of communication link. One of those two is selected depending on polarity of previous data transmitted. The developed program had to be split into different program blocks because of different data conversion and clocking cycles as described in chapter 4. The blocks capture the incoming data from the RTDS and store them onto a custom memory. Further on another block read the data back and encodes them to 8 bit sequences to meet the requirements for the Aurora 8B/10B protocol. After the data got transferred and the result gets back, these data sequences gets converted back to 32 bit values and stored again in a different part of the custom memory. Now the data can be read by a transfer block which reads the data from the memory and transfers it in a certain order back to the RTDS. The protocol was implemented on a Xilinx ML 605 FPGA evaluation board and successfully tested in a CHIL experiment which uses a custom controller board (HCB) for voltage source conversion and novel design emphasis flexibility. A pure digital communication way with an industrial based standard protocol is now possible and should be further developed in the future.

The next step in the development of this topic could be the implementation of an upper level communication protocol with Herb´s controller board to test in more detail the data transfer of more values with different sizes and to outline the daily use for HIL and CHIL testbeds at CAPS.

6 LIST OF FIGURES

7 BIBLIOGRAPHY

[1] B. Wagner, "All-Electric Ship Could Begin to Take Shape By 2012," National Defense Magazine, Archive Nov. 2007.

[2] P. B. Backlund, C. C. Seepersad, and T. M. Kiehne, "A System-Level Design Framework for All-Electric Ship Thermal Systems," American Society of Naval Engineers, 2012.

[3] N. H. Doerry and J.C. Davis, "Integrated Power System for Marine Applications," Naval Engineers Journal, May 1994.

[4] T.J. McCoy, "Trends in ship electric propulsion," Power Engineering Society Summer Meeting, 2002 IEEE, vol. 1, pp. 343–346, Jul 2002.

[5] G. Buja, A. da Rin, R. Menis, and G. Sulligoi, "Dependable Design Assessment of Integrated Power Systems fo All Electric Ships," Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS),, pp. 1–8, 19-21 Oct. 2010.

[6] Capt. N. Doerry. Open Architecture Approach for the Next Generation Integrated Power System, 2013, Online available: https: //www.navalengineers.org/SiteCollectionDocuments/2007%20Proceedings%20Documents/Automation%20and%20Controls%202007/1-2-9%20Paper.pdf

[7] L. HanYu and M. LongHua, "Integrated Protection on Shipboard Integrated Power System," The International Conference on Advanced Power System Automation and Protection (APAP), vol. 1, pp. 252–255, 16-20 Oct. 2011.

[8] V. R. Basam and A. Das, "All Electric Ship-The Super Platform for Tomorrow's Naval Warfare," Second International Seminar And Exhibition on Naval Armaments (NAVARMS), 2010.

[9] K. P. Logan, "Intelligent Diagnostic Requirements of Future All-Electric Ship Integrated Power System," IEEE Transactions on Industry Applications, vol. 43, no. 1, pp. 151–163, Jan. 2007.

[10] N. Doerry, “Next Generation Integrated Power Systems for the Future Fleet,” IEEE Electric Ship Technologies Symposium, Baltimore, Apr. 2009.

[11] B. Jacobson and J. Walker, “Grounding Considerations for DC and Mixed DC and AC Power Systems,” Society of Naval Engineers Journal, 2007, vol. 119, no. 2, pp. 49–62, 2007.

[12] IEEE STD. 45, “IEEE Recommended Practice for Electrical Installations on Shipboard,” pp. 1–91, 2011.

[13] IEEE STD 142 (Green Book), “IEEE Recommended Practice for Grounding of Industrial and Commercial Power Systems,” 1982, IEE Standards Association - Power System Engineering Committee and IEEE Industry Applications Society

[14] M. Kofler, “Novel Approach to Analyze Transients in Shipboard Power Systems Based on Scattering Parameters Modeling,” Research Paper, 2012, CAPS Florida State University

[15] H.W. Dommel,”Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks,”IEEE Transactions on Power Apparatus and Systems, April 1969, vol. PAS-88, no. 4, April 1969, pp. 388-399.

[16] Electromagnetic Transient Program (EMTP) Rule Book, Bonneville Power Administration, 1995

[17] O.Nayak, G.Irwin, A.Neufeld,”GUI Enhances Electromagnetic Transients Simulation Tools,” IEEE Computer Applications in Power, January 1995, January 1995, pp. 17-22.

[18] J.R.Marti, L.R.Linares,”Real-time EMTP-based Trausients Simulation,” IEEE Transactions on PowerApparatusandSystems, vol. 9,110.3, August 1994,pp. 1309-1317

[19] M. Kezunovic, M. Aganagic, V. Skendzic, L. Domaszewicz, J.K. Bladow, ’Transients Computation for Relay Testing in Real-time,”IEEE Transaction on Power Delivery, July 1994, vol. 9, no. 3, pp. 1298-1307.

[20] R. Kuffel J. Giesbrecht T. Maguire R.P. Wierckx P. McLaren,"RTDS-a fully digital power system simulator operating in real time," 1995 IEEE, pp. 498-502.

[21] R.Kuffe1, P.McLaren, M.Yalla, X.Wang, 'Testing of the Beckwith Electric M-0430 Multifunction Protection Relay Using a Real-Time Simulator (RTDS)," April 1995

[22] H.Duchbn, M.Lagerkvist, R.Kuffe1, R-Wierckx "HVDC Simulation and Control System Testing Using a Real-Time Digital Simulator (RTDS)," April 1995, College Station, USA.

[23] X.Wang, J.Giesbrecht, D.Woodford, L.Arendt, R.Wierckx, R.Kuffe1,"Enhanced Performance of a Conventional HVDC Analogue Simulator with a Real-Time Digital Simulator," 1993, Vol. 1, pp. 663-669.

[24] R.Kuffe1, R.Wierckx, P.Forsyth, H.Duchkn,M,Lagerkvist,X.Wang, "Expanding an Analogue HVDC Simulator's Modelling Capability Using a Real-Tie Digital Simulator (RTDS)," April 1995, College Station, USA

[25] M. Sloderbeck, C. Edrington, M. Steurer,"Hardware-in-the-Loop Experiments with a Simulated Electric Ship Power System Utilizing a 5 MW Variable Voltage Source Converter Amplifier," 2009 IEEE, pp. 1170-1187

[26] Michael Steurer, Richard Meeker, Karl Schoder, and Peter McLaren, "Power Hardware-in-the-Loop: A Value Proposition for Early Stage Prototype Testing," 2011 IEEE , pp.3731-3734

[27] Hardware-in-the-loop (HIL) simulation, 2013 Online Available: http://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=download&collection_id=dadf2f8757873d74&writer=rl&return_to=Hardware-in-the-loop+simulation

[28] Getting started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit, 2011 Online Available: http://www.Xilinx.com/support/documentation/boards_and_kits/ug533.pdf

[29] FM-S18, 2103, Online available: http://www.xilinx.com/products/boards-and-kits/1-SHZI2.htm

[30] Aurora 8B/10B Protocol Specification, 2010, Online available: http://www.Xilinx.com/support/documentation/ip_documentation/aurora_8b10b_protocol_spec_sp002.pdf

[31] Xilinx LogiCORE IP Aurora 8B/10B v6.2 User Guide, 2010, Online available: http://www.Xilinx.com/support/documentation/ip_documentation/aurora_8b10b_ug766.pdf

[32] Xilinx Virtex-6 FPGA GTX Transceivers User Guide, 2011 , Online available: http://www.Xilinx.com/support/documentation/user_guides/ug366.pdf

[33] Xilinx ML 605 Hardware User Guide, 2011, Online available: http://www.Xilinx.com/support/documentation/boards_and_kits/ug534.pdf

[34] Homepage Xilinx ISE Design Suite, 2013, Online available: http://www.xilinx.com/

[35] Homepage RTDS Technologies, 2013, Online available: http://www.rtds.com/index/index.html

[36] Xilinx Core Generator System, 2013, Online available: http://www.xilinx.com/tools/coregen.htm

[37] Mahmoud Matar, Houshang Karimi, Amir Etemadi, Reza Iravani,“ Performance Real-Time Simulator for Controllers Hardware-in-the-Loop Testing”, mdpi energies journal 2012, p 1714

[38] Homepage Idaho National Laboratory, 2013, Online available: https://inlportal.inl.gov/portal/server.pt/community/national_and_homeland_security/273/modeling_and_simulation/1707

8 ANNEX

A ARCHIVING

The research presented in this thesis was performed in 2013 at the Center for Advanced Power Systems (CAPS). As a result of the research a range of documents, scripts and simulation files were created. This appendix gives an overview of what is available, so students and ESRDC members can use the result for further investigations. All the files which have been created while working on this project are stored in the student directory on the PLASMA-Server at CAPS. The full path to access the files is \\plasma\STUDENT\berger.

The folders in this directory are described in the attached READ_ME.txt files which also serve as a walkthrough of the steps needed to use each program and their corresponding files.

1_Working program: The programs to use on RTDS, ML 605 and Herb´s Controller Board to reproduce the current status of the project. Also the raw Rocket I/O core is included to rebuild the program from scratch or to add another one for more transfer possibilities over the fiber ports.

2_Docs: A directory, including all files which were necessary and helpful to complete the project such as literature, datasheets, specifications, drawings, orders, etc.

B USED PROGRAMS

Description:

Xilinx ISE (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. The tool was used to create and compile the VHDL code to be used on the evaluation board (see Fig. 8.1 below) [34].

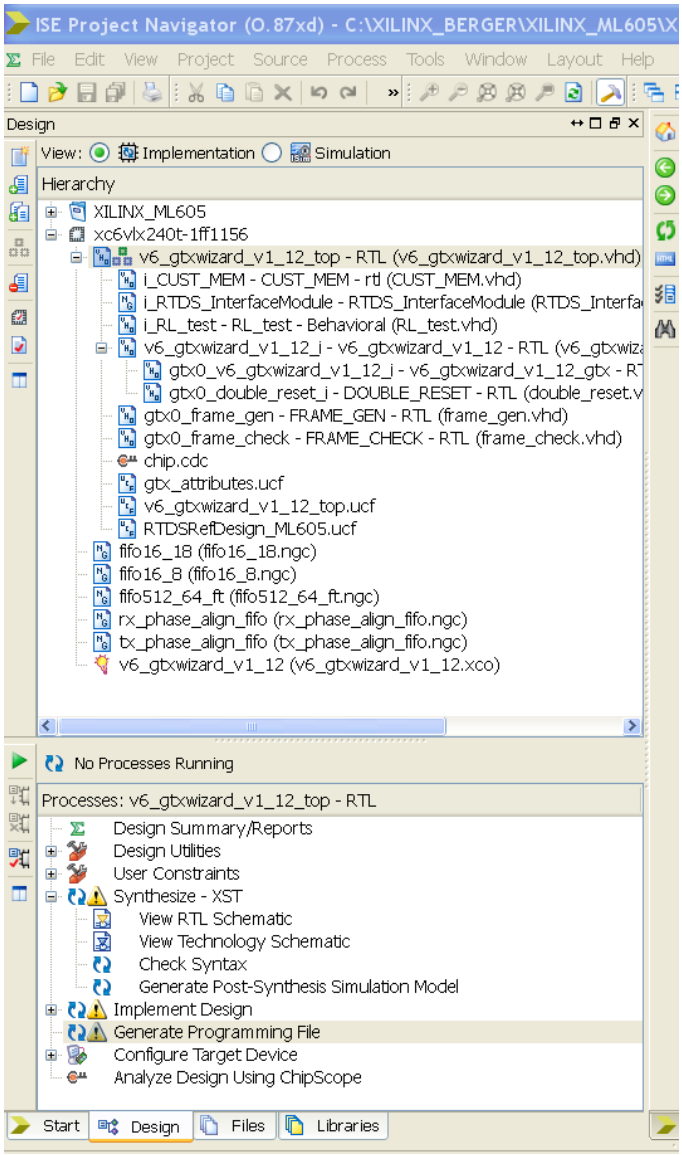


Figure 8.1: Xilinx ISE Design Suite 13.4

How to Use:

1. Click on *"File"* → *"Open"…* → go to the directory containing *"v6_gtxwizard_v1_12_top.vhd"* and open. This is the main program which makes all the connections to the physical pins on the evaluation board and the virtual blocks which are the logic of the program.

2. Press the button *"Generate Programming File"* (left bottom) to create the .bit file which is used to program the evaluation board which iMPACT (see next chapter)

3. To edit the blocks just *double click* on them and the code appears on the right side of the ISE Design Suite Tool.

4. A complete graphical overview is shown when you double click *"View RTL Schematic"* (left bottom…expand Synthesize – XST). This overview shows the different signals between the blocks and their relation to each other. Also you can easily check if every connection is made and the right inputs and outputs are connected.

5. The created *"chip.cdc"* file allows you also to *"Analyze Design Using Chipscope"* (very left bottom), when you double click here the code gets compiled and automaticly Chipscope analyzer gets opened to program the evaluation board directly from the Chipscope program. This step also allows you to verify the input and output data at each connection of a block or signal to see if the whole program runs as expected.

6. Verifying the code either by generating the programming file or analyzing it with chipscope can take several minutes. Warnings during the compilation can be normally ignored because they have no effect on the functionality of the code.

C IMPACT

Description:

The Xilinx iMPACT programming and file generation software (see Fig. 8.2) is used to create a valid PROM file for the Platform Flash PROM family. The basic flow takes an FPGA design bitstream input and converts it into a valid PROM file format (MCS/EXO). Designers accessing the advanced features of the XCFxxP device must generate the CFI file. This file enables the programming of the XCFxxP design revisions, CLKOUT, or decompression features [34].

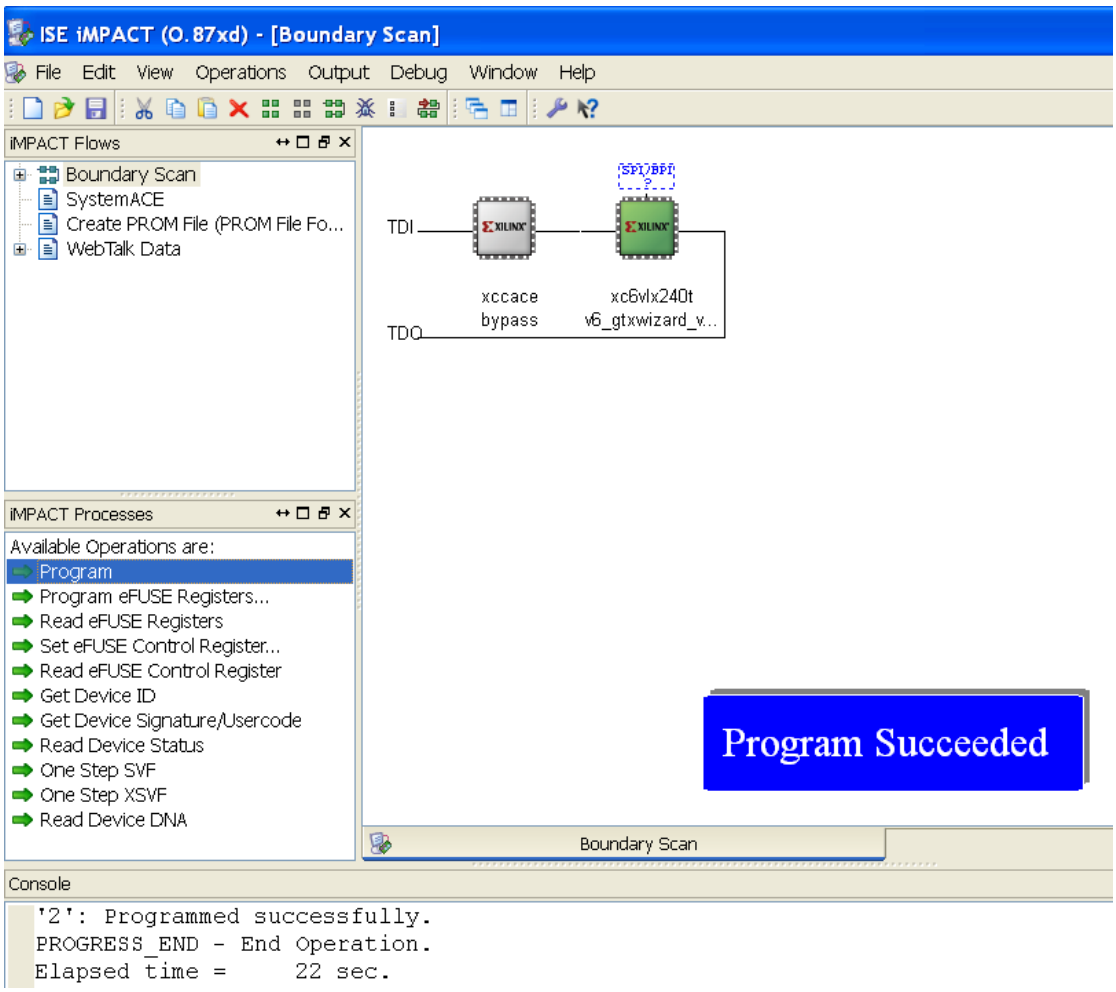


Figure 8.2: ISE iMPACT

How to Use:

1. Double click on *"Boundary Scan"* → *"Initialize Chain"* appears in the middle of the right section, after double clicking here the program enables a connection to the evaluation board and registers all chips which can be programmed and their relations to each other.

2. As an example in Fig. 12.2 the name of the Virtex 6 chip appears as "xc6vl240t", right click on that chip and *"Assign new programming file"*. The program .bit file can be assigned directly because the flash memory of the chip is accessible.

3. Go to the right directory and chose the .bit file of your program. After the program has verified the .bit file you can program the chip.

4. Either click on *"Program"* (left bottom) or right click on the chip again and click *"Program chip".*

5. After the evaluation board was successfully programmed a blue box appears with the message *"Program Succeeded"*. Right after that you should check the connection signal LED on the evaluation board which shows you if a connection were established between the RTDS and the board. It can happen that no connection is established (LED low) at the first attempt so just repeat step 4 until connection is established.

6. Now the program is up and running until the board loses power.

Exception:

(Herb´s controller board with a Virtex 4 FPGA chip)

1. Double click on *"Boundary Scan"* → *"Initialize Chain"* appears in the middle of the right section, after double clicking here the program enables a connection to the evaluation board and registers all chips which can be programmed and their relations to each other.

2. Because the flash memory of the chip cannot be accessed directly you have to program the PROM with iMPACT and to do so you have to generate a PROM-file out of the .bit file. For more instruction see (Platform Flash PROM User Guide by Xilinx Chapter 6: PROM File Creation and Programming Flow)

3. After successfully flashing the PROM you have to power down the board and restart it again. Wait a few seconds until the board is completely powered up and the current consumption is about 1A. Press the *"RESET button"* on the board so the program from the PROM gets flashed automatically onto the Virtex 4 chip.

4. The new program stays in the PROM even after the board loses power so every time you power up the board the Virtex 4 FPGA chip gets programmed with the assigned program from the PROM.

D RSCAD

Draft

Description:

Users can graphically layout a schematic diagram of the system to be simulated using the RSCAD/Draft module (see Fig. 8.3). The right side of the Draft screen contains various libraries of power and control system components. To create a schematic, icons are simply taken from the library and placed onto the “Draft Canvas” on the left side of the screen. Then the required parameters associated with that model are entered in a pop-up window. Features such as group/ungroup commands are provided for easy on-screen arrangement. The compile process provides preliminary error checking of component parameters and generates the real time code that will be run on the RTDS Hardware [35].

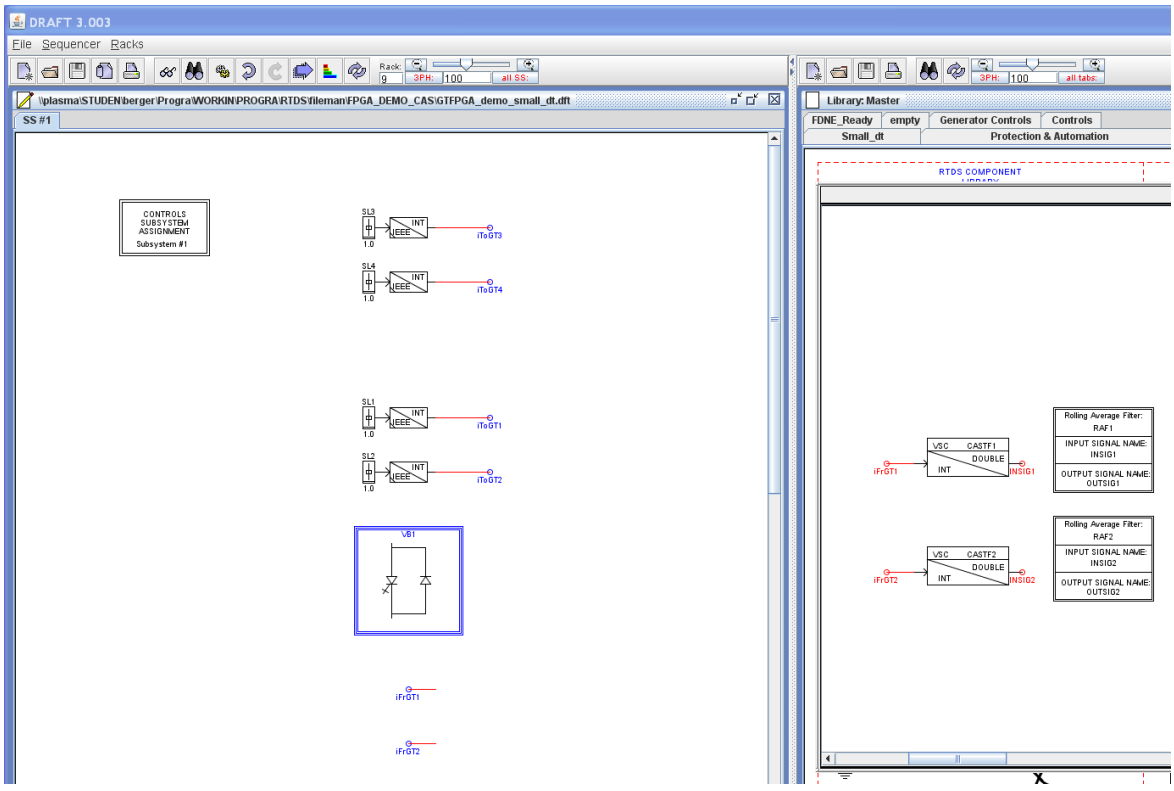


Figure 8.3: DRAFT 3.003

RunTime:

Description:

All loading, running, and controlling of the simulation is done entirely from the host computer through the RSCAD RunTime module (see Fig. 8.4). This module is also known as the Operator's Console. The RunTime screen is customizable for each simulation by creating meters, plots, sliders, buttons, dials, switches, etc. The user is able to control and interact with the simulation in this graphical environment. Non-critical monitoring can be minimized for efficient use of workspace. RunTime automatically triggers plot updates during simulations. Calculation functions are available to condition plots. As well, text, annotation and time stamps can be added to categorize simulation result. Plot data can be saved for post processing in MultiPlot, printed, and or saved in pdf, jpeg, and vector format [35].

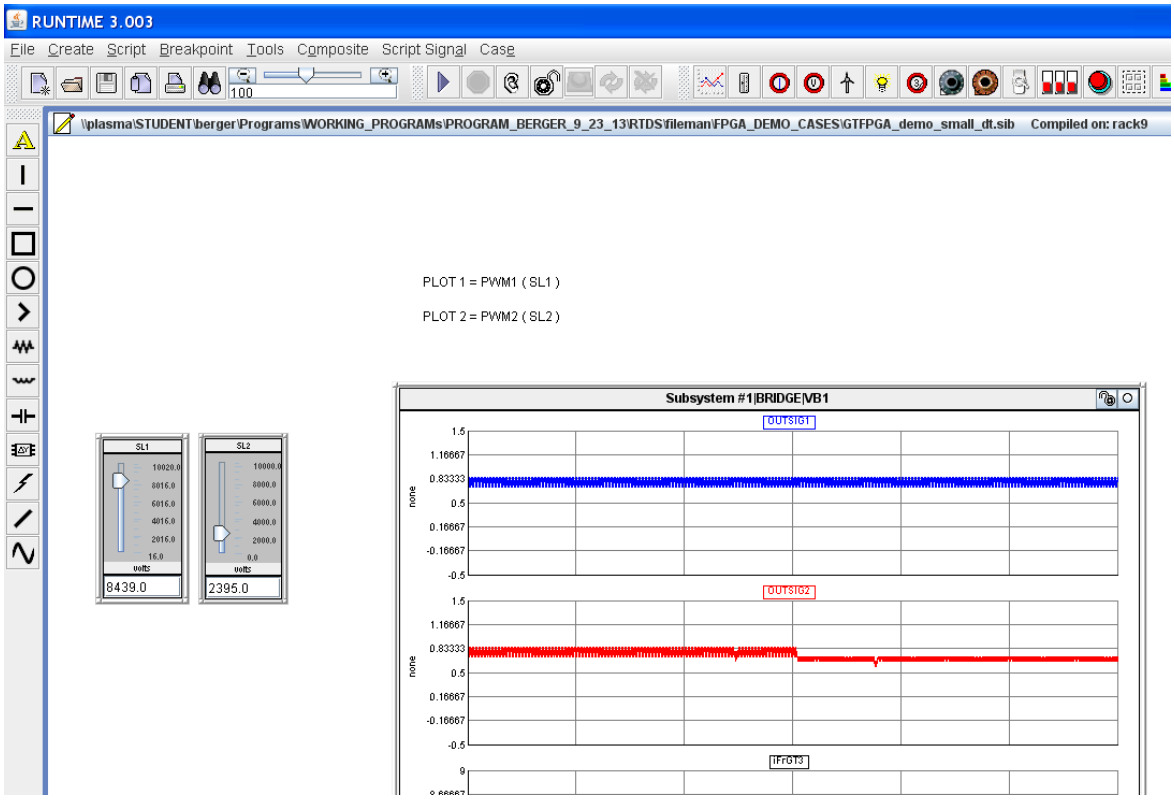


Figure 8.4: RSCAD 3.003 RUNTIME

How to Use:

1. Open the RSCAD file manager and change the *"Shared Lock Directory"* to \\plasma\caps\software\rscad if necessary (see Fig. 8.5).

2. Also change the *"RSCADUser Directory"* to: \\plasma\STUDENT\berger\Programs\WORKING_PROGRAMs\PROGRAM_BERGER_9_23_13\RTDS (see Fig. 8.5).

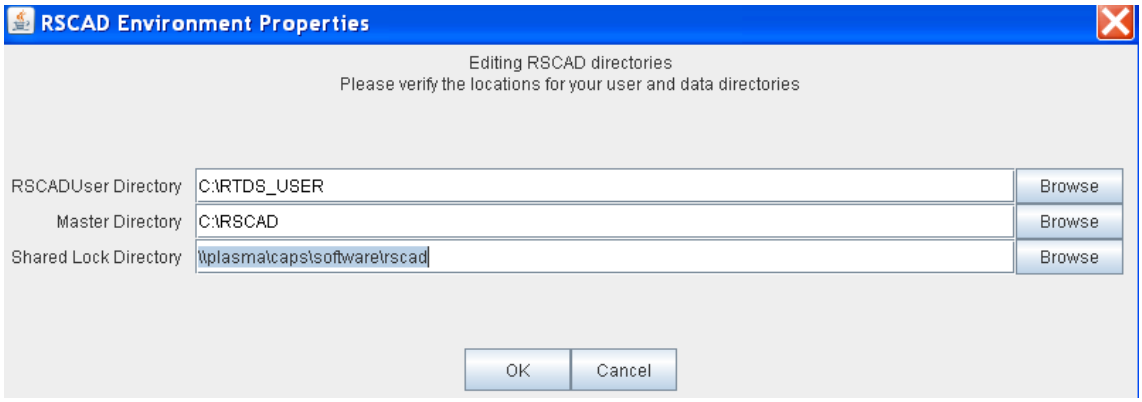


Figure 8.5 Changing the directory

3. Double click on the file *"GTFPGA_demo_small_dt.sib"* and the RUNTIME opens, also double click on *"GTFPGA_demo_small_dt.dft"* to open the DRAFT.

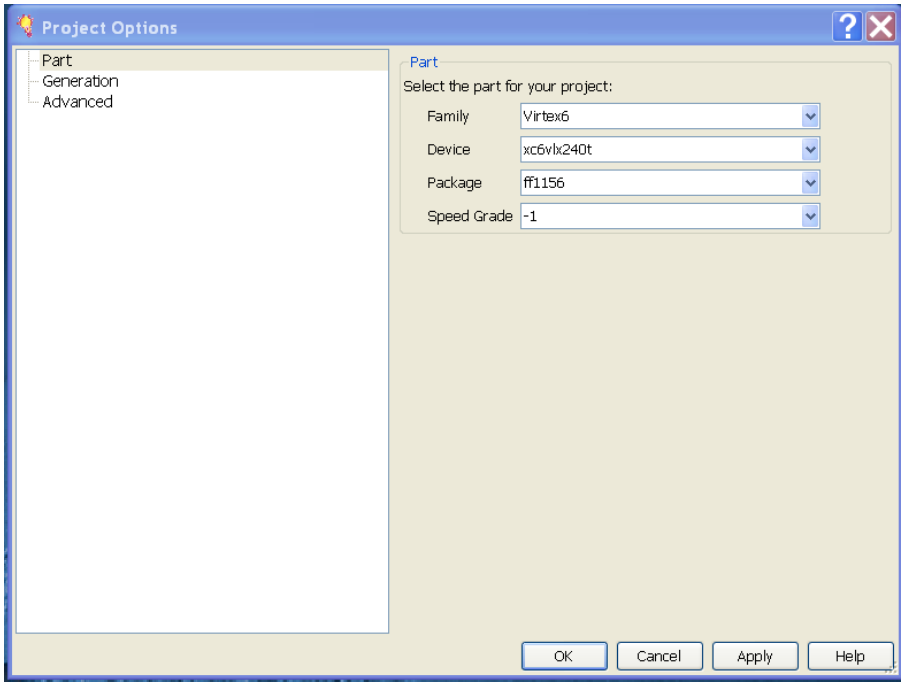
E XILINX CORE GENERATOR

Description:

Xilinx CORE Generator™ System accelerates design time by providing access to highly parameterized Intellectual Properties (IP) for Xilinx FPGAs and is included in the ISE® Design Suite. CORE Generator provides a catalog of architecture specific, domain-specific (embedded, connectivity and DSP), and market specific IP (Automotive, Consumer, Mil/Aero, Communications, Broadcast etc.). These user-customizable IP functions range in complexity from commonly used functions, such as memories and FIFOs, to system-level building blocks, such as filters and transforms. The following walkthrough shows you how to generate the needed Aurora 8B/10B core with the exact specifications [36].

How to Use:

1. Open Xilinx CORE Generator > New Project … create New Folder named Coregen >save
2. When Project Options window opens use following settings and then hit OK:



3. In core generator > IP catalog > view by function >FPGA features and design > IO Interfaces >(double click on) Viertex-6 FPGA GTX Transceiver Wizard.

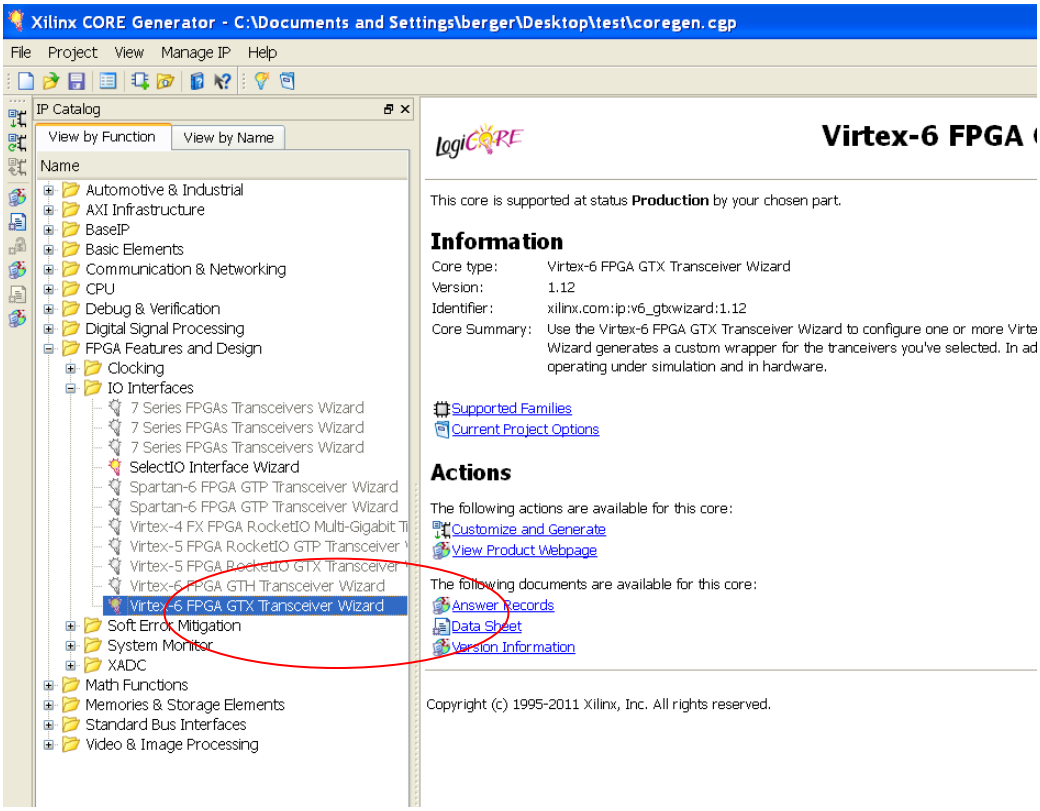


Figure 8.6: Xilinx CORE Generator™

4. In the opened window type appropriate Component Name(or just keep as it is displayed ”rocketio_wrapper”) .

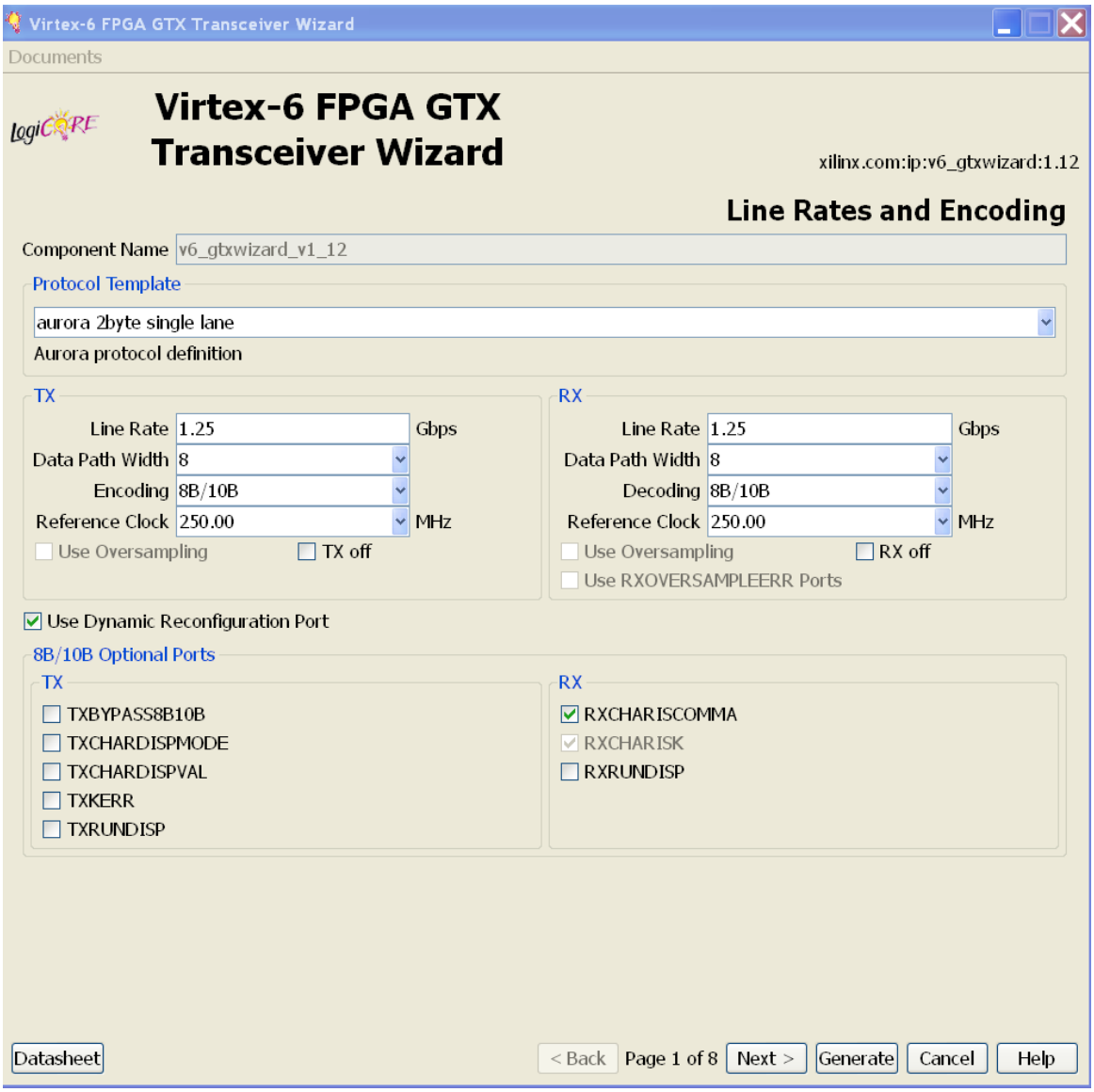


Figure 8.7: Virtex-6 FPGA GTX Wizard, page 1

5. With the help of following screenshots the RocketIO MGT core can be generated. This software core (bunch of VHDL and other project files) will then be needed to add in the Main project of the Xilinx ISE.

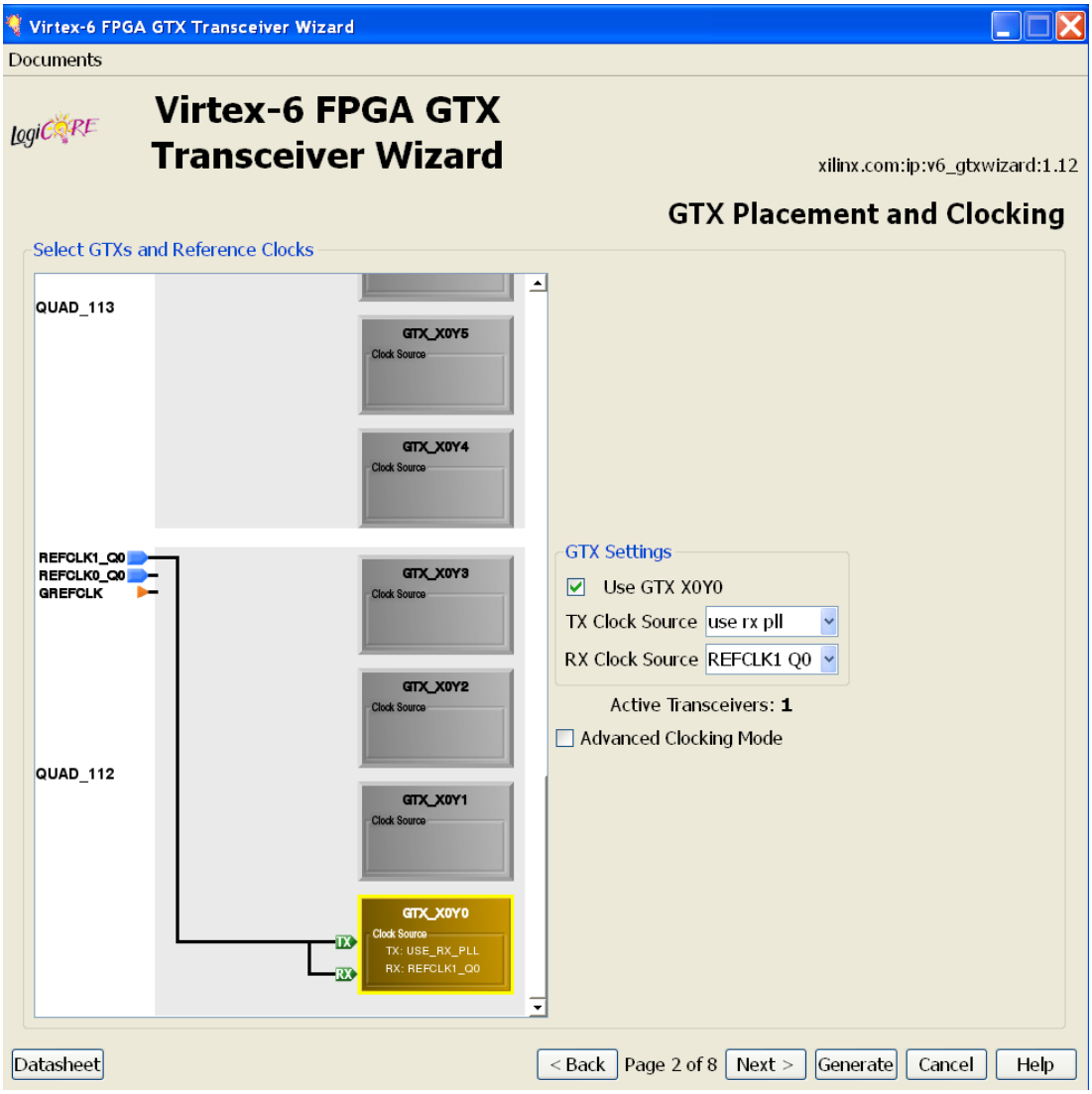


Figure 8.8: Virtex-6 FPGA GTX Wizard, page 2

GTX_X0Y0 = Channel 4 on FM-S18

GTX_X0Y1 = Channel 5 on FM-S18

GTX_X0Y2 = Channel 6 on FM-S18

GTX_X0Y3 = Channel 7 on FM-S18

GTX_X0Y4 = Channel 0 on FM-S18

GTX_X0Y5 = Channel 1 on FM-S18

GTX_X0Y6 = Channel 2 on FM-S18

GTX_X0Y7 = Channel 3 on FM-S18

GTX_X0Y17 = SFP connector and cage (see file: ug534, ML605, page 13, number 10) onboard

Almost DFAULT

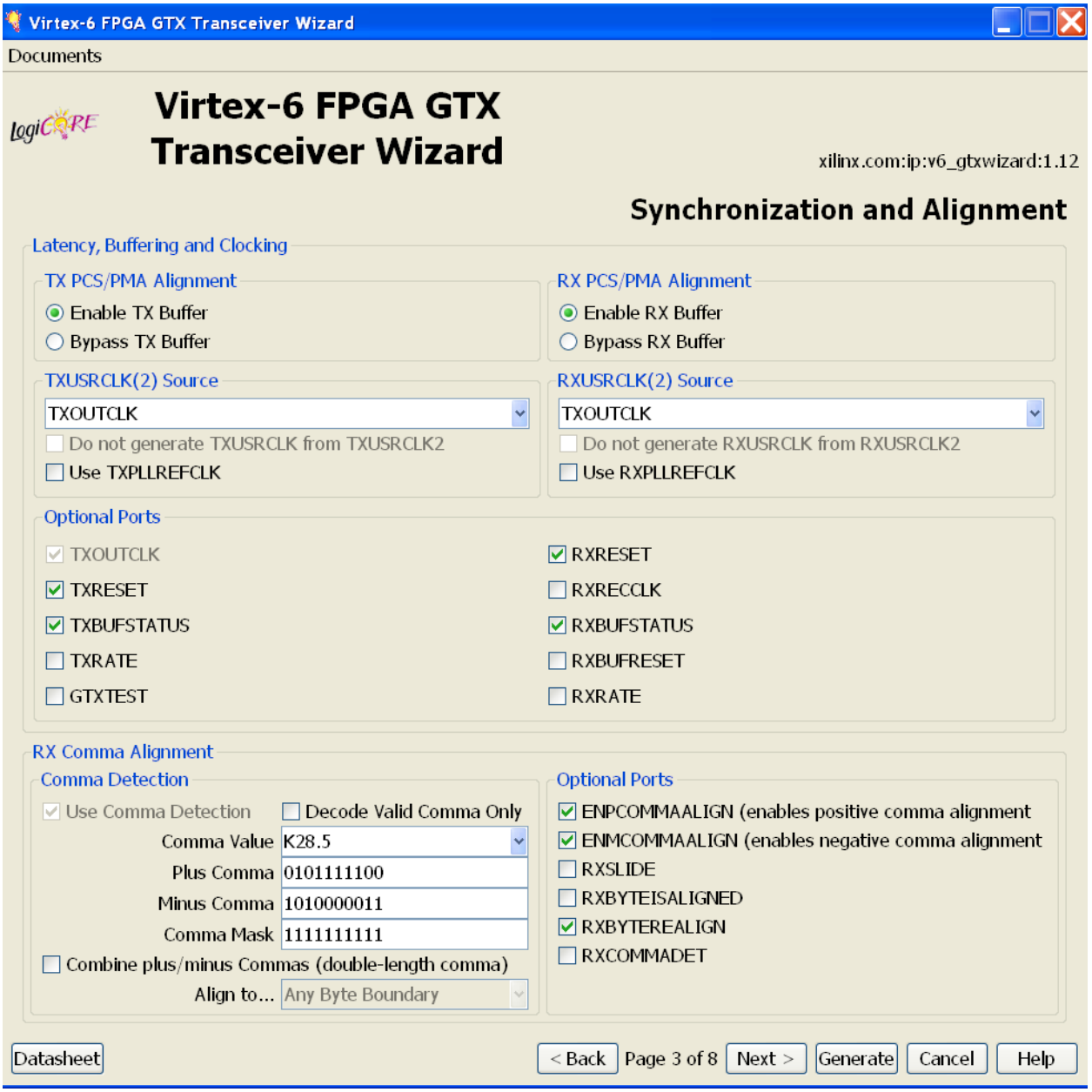


Figure 8.9: Virtex-6 FPGA GTX Wizard, page 3

Almost DEFAULT

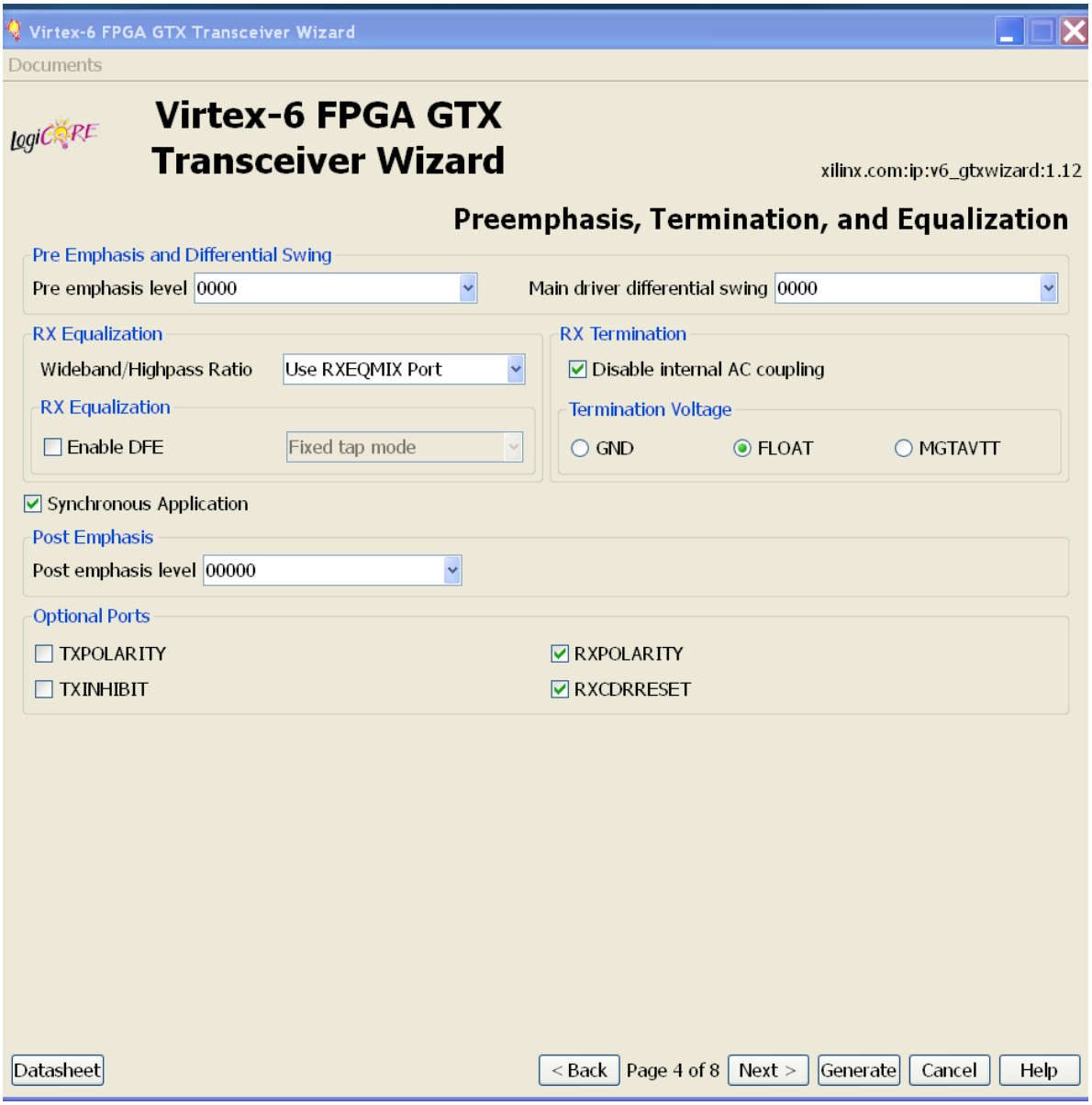


Figure 8.10: Virtex-6 FPGA GTX Wizard, page 4

DEFAULT

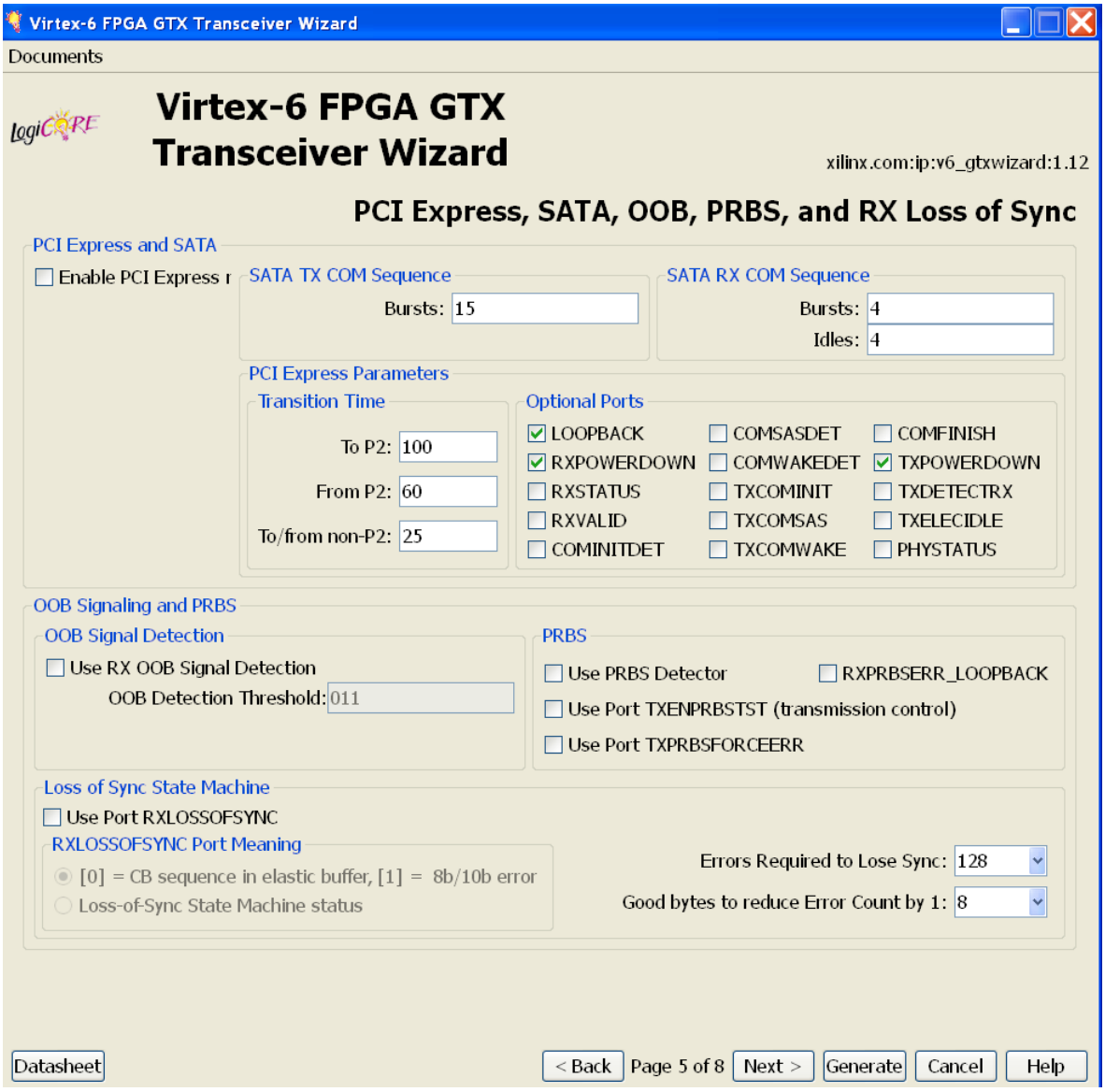


Figure 8.11: Virtex-6 FPGA GTX Wizard, page 5

DEFAULT

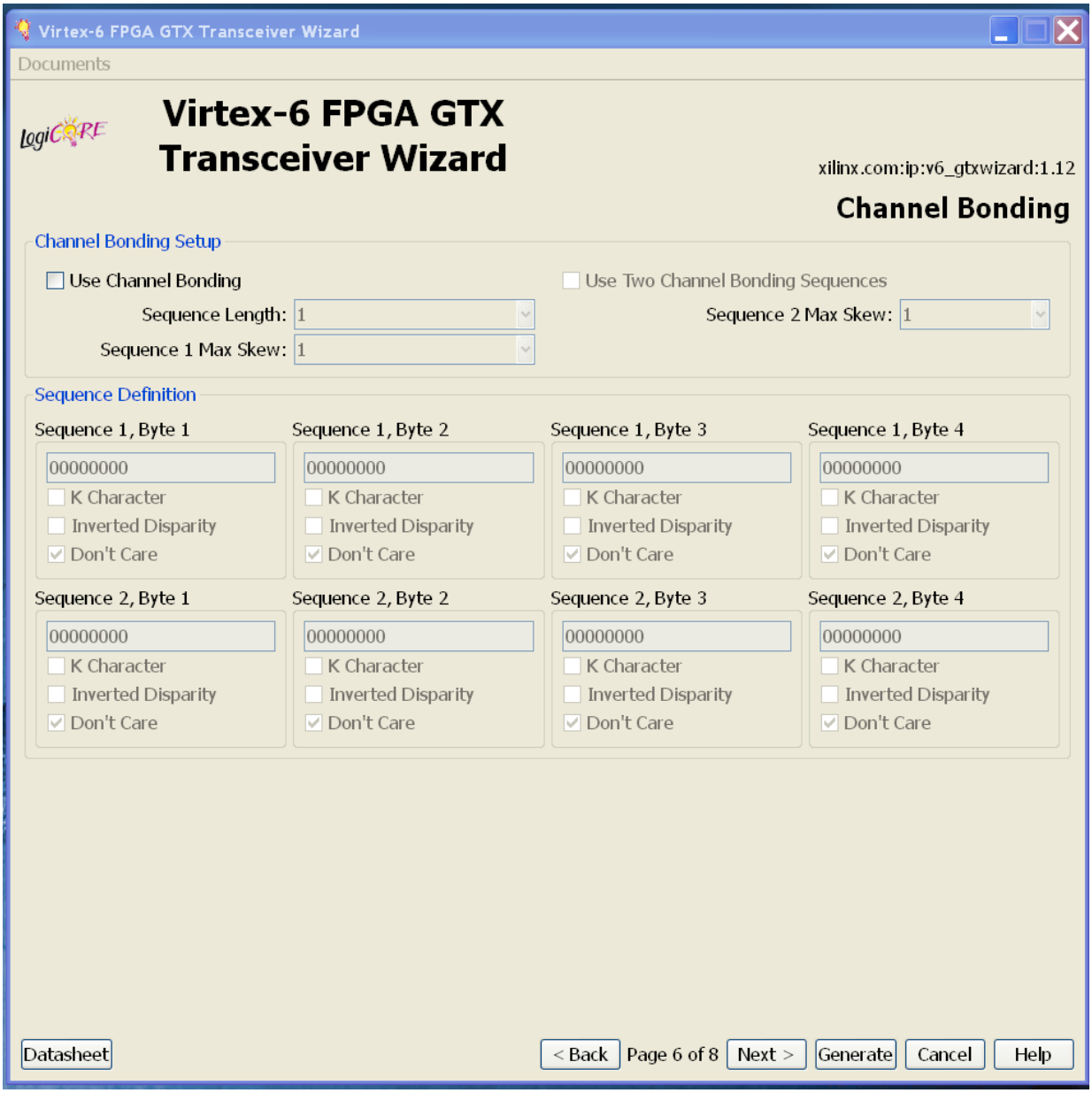


Figure 8.12: Virtex-6 FPGA GTX Wizard, page 6

Figure 8.13: Virtex-6 FPGA GTX Wizard, page 7

DEFAULT

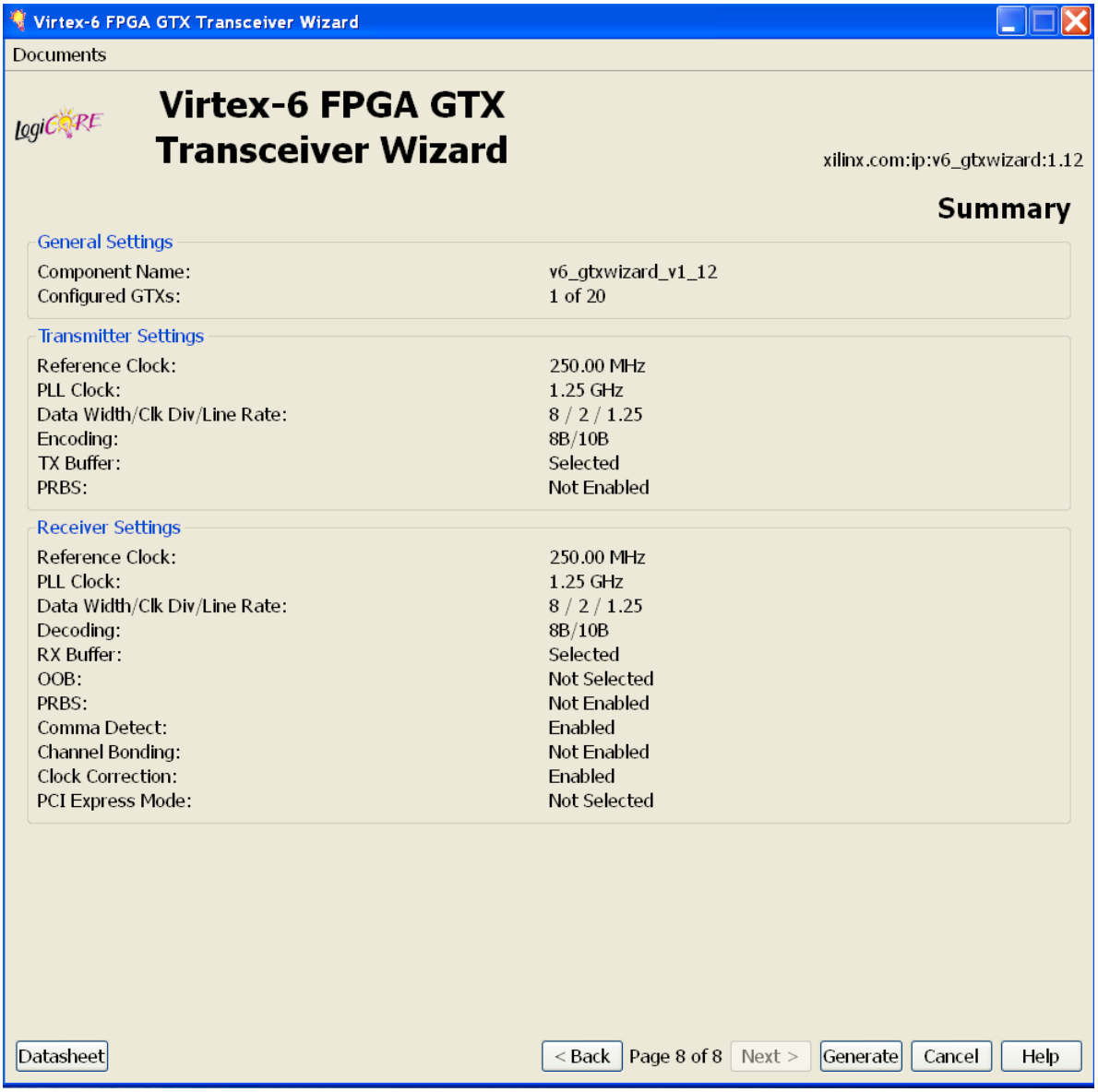


Figure 8.14: Virtex-6 FPGA GTX Wizard, page 8