N°3

ECOLE POLYTECHNIQUE

UNIVERSITAIRE DE MONTPELLIER

**End of studies industrial project Electronics, Robotics, Industrial Computing Department**

*YEAR 2022-2023*

**PROJET**

**Camille HERRMANN**

**FPGA IN THE LOOP**

A picture containing text, metalware, gear

Description automatically generatedECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER

UNIVERSITE MONTPELLIER II SCIENCES ET TECHNIQUES DU LANGUEDOC

Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5

Logo

Description automatically generatedTél. : 04 67 14 31 60 – Fax : 04 67 14 45 14

E-mail : [scola@polytech.univ-montp2.fr](mailto:scola@polytech.univ-montp2.fr)

Table of contents

Contents

[Introduction: 3](#_Toc125702064)

[PFE and test on FPGA 3](#_Toc125702065)

[Company and project idea 3](#_Toc125702066)

[Project blueprint 5](#_Toc125702067)

[Specification 5](#_Toc125702068)

[The process: tools, methods 5](#_Toc125702069)

[The Simulator 5](#_Toc125702070)

[The generic test design 6](#_Toc125702071)

[The board 6](#_Toc125702072)

[Work done 7](#_Toc125702073)

[First steps with the tools 7](#_Toc125702074)

[Quartus and the board 7](#_Toc125702075)

[MATLAB 8](#_Toc125702076)

[Adapter design and test 10](#_Toc125702077)

[Raw design on Quartus 10](#_Toc125702078)

[Design in FIL 13](#_Toc125702079)

[Test the design with a filter 17](#_Toc125702080)

[Conclusion 19](#_Toc125702081)

[Feedback on results 19](#_Toc125702082)

[Notions learned and competences 19](#_Toc125702083)

[What next 20](#_Toc125702084)

[Acronyms 21](#_Toc125702085)

[Useful links 22](#_Toc125702086)

# Introduction:

## PFE and test on FPGA

This report retraces my End of Studies Project (PFE), also called Industrialization Project. It takes place at my last year of engineering school and represents around 300 hours. The objective of the project is to put students in charge of finding solutions to concrete problem. This is done within the School at Polytech Montpellier on behalf of an industrial partner: General Electric[[1]](#footnote-1) in my case.

The Project’s main domains are Field Programmable Gate Arrays (FPGA) and their tests. FPGA are becoming more important in industry notably for two main assets: a reconfigurable architecture and parallel processing. The reconfigurable architecture allows a high level of flexibility and adaptability of design systems. Parallel processing means that multiple tasks can be executed simultaneously which implies faster computation. The parallel processing is due to the architecture composed of configurable logic blocks and interconnections.

As every electronic technology, before commercialization, the product or service needs to be tested. FPGA testing are usually performed by simulation or prototyping. Simulation means to use only software to simulate the behavior of the IP to be tested, usually before implementing it on the board. It’s useful for a simulation at one cycle close. It means to look with short timing range around the nano or microsecond, to control in detail the signals or timing. Finally, a drawback that led to the project’s idea is that it is not an applicative solution to run on longer period or real timing scale application. To run a simulation around the minute requires long time and powerful computer. Prototyping involves to physically implement the IP on a development board allowing to test on actual hardware. The results are applicative since it’s run on the same hardware that will be used. However, prototyping involves more resources and complexity to run tests and limited tools for debugging in case of any errors.

## Company and project idea

General Electric (GE) is a multinational corporation that operates in several key industries, including power, renewable energy, healthcare, and aviation with a presence in over 180 countries, GE is a leader in providing innovative solutions to a wide range of customers. In France, GE has a significant presence with the global headquarters of GE Renewable Energy in Montpellier. The Montpellier office also houses the global headquarters of GE Grid Solutions, one of the three subdivisions of GE Renewable Energy. Grid Solutions provides power utilities worldwide with equipment, systems, and services to bring power reliably and efficiently through substations. This includes advanced software and hardware technologies for energy management, distribution, and transmission, with the goal of improving the reliability and stability of the grid. This office is also the headquarters of GE Renewable Energy for the Mediterranean region.

It is the R&D team of the Montpellier Grid Solutions of Renewable Energy GE that proposes a project around the test of the FPGA IP. The R&D team is specialized in control-command for very high voltage in electricity distribution. The aim of the current project is to test their IPs in real time, which would reduce the time needed for test. In addition, a major goal is to create a generic benchmark around the Avalon Streaming (AST) interface, an internal communication protocol specifically designed for use within Intel's FPGA technology. The benchmark will take the IP to be tested as a component so that many IPs could be tested without having to design new tests for each IP. The MathWorks tool “FPGA in the loop” (FIL) represents a solution. FIL allows to use a large range of libraries or mathematics tools through MATLAB features therefore to work on applicative test, in real time. Also, FIL facilitate the integration of signal processing and control loop into test chain. FIL create a Simulink block containing the binary executable to send to the FPGA SoC. Simulink, on the computer, simulates signals and test’s environment (such as a closed loop) while the board executes calculations and sends back the results.

Diagram, schematic

Description automatically generated

Figure 1: FPGA In the Loop process

# Project blueprint

## Specification

As said before the main idea is to test IP as fast and efficiently possible. For the timing constraint tests are going to be run in real time, the test environment is a simulation on a software simulator, but the calculation of the IP is done on a board in real time. So, it’s a mix between simulation and prototyping methods.

We consider here that the IPs to be tested are all working on the Avalon interface, more precisely the AST. Using this assumption, we want to build a test design ready to simulate and test different IPs that are inserted in the design.

Simulator

Figure 2: schematic of the three main part of the project, Simulator-Testbench-Board

Board

Generate the AST signals and interface with the IP

Create the input data

IP tested in real time

observable

results

Receives the results and sends to simulator

So, the specifications are to be able to run a mix simulation/prototype test in a FPGA development board including a generic testbench as a wrapper between the simulator with raw data and the IP on the board on AST interface.

## The process: tools, methods

### The Simulator

The simulator needs to be able to create signals and models but also to provide a real time processing. The tool from MathWorks called FPGA-In-the-Loop (FIL) allows these two needs. FIL is a simulation technique that allows to test and verify the behavior of an IP running on an FPGA board. The test is run using Simulink, a graphical programming environment for modeling, simulating, and analyzing dynamic systems. So, we can use the assets of Simulink and those of a board to be faster and efficient:

* Using a board enables the implementation IP on the actual hardware and interact in real time so the simulation can be faster and more precise rather than just simulating in software. The test in real time allow to quickly identify an issue and speed the development process.
* Simulink allows to design and test an IP in a graphical environment therefore to see the behavior of the design in real time. Any adjustments can be done quickly to answer to our needs. In addition, on Simulink are a large variety of tools and features to develop, automatize tests and to debug notably around signal processing.

### The generic test design

This part of the project is about designing and developing a kind of adapter / connecter between the simulator and the board. The simulator provides raw data, while the IP to be tested has AST inputs and outputs (I/O). Avalon is an interface architecture for FPGA design developed by Intel. The idea is to simplify complex design by transmitting data with simple transfer data protocol. The AST is flexible on the data width and control signals.

The goal of the adapter is to create the AST interface environment needed for the IP depending on the data from the simulator and to deal with the different clocks. Indeed, the Simulator has low frequency because it is working on real time, but the board is working at high frequency: 125 MHz. This adapter will be developed in VHDL on Quartus environment.

### The board

One condition we have on the board is that it needs to have an 1G/b ethernet port to communicate with the simulator seen above. So, I’m using the Cyclone V GX FPGA board from Intel. About the development, to use the cyclone V GX I’m using Quartus Prime lite also from intel and its tools. Precisely on Quartus I’m using the design tools such as:

- The IP catalog: list of designed IPs that can be used and changed (such as memory blocks)

- ModelSim Altera Simulator: tool to simulate and debug designs allowing to observe waveforms.

- Signal Tap logic analyzer: to observe internal signals of the design while it’s running on the board.

# Work done

## First steps with the tools

### Quartus and the board

First, I fetched the board documentation and then download the tools needed. The Cyclone V GX board is supported by the Quartus prime software development. I downloaded Quartus 18.1 standard lite because it doesn’t require any licenses and it supports Cyclone board family.

To manage the board, I started to create a simple project to use the tools that are needed for later. First, I wrote a code that change the state of a led when a button is pushed. So, I could work with inputs and outputs of the board and the pin assignment tool. Then to use the IP catalog management I created a PLL (Phase Locked Loop) to have a clean clock signal and a RAM memory to store timing information when the button is pushed [[2]](#footnote-2). Using it, I then used ModelSim Altera tool. To use ModelSim I wrote a testbench. The testbench is a software file used to provide inputs to the FPGA. The simulation of the testbench allows to simulate the behavior of the design. Once the testbench wrote and ready, I can observe waveforms of the I/O and signals of each level of the file hierarchy.

Graphical user interface, text

Description automatically generated

Figure 3: Simulation of simple code, led changes when button is pushed

The picture above shows the results of the simulation on ModelSim graphic interface. The left section is a tree structure of the components in the testbench called “tb\_btn\_memory.” When I click on an element of the tree structure I can see on the middle section “Objects” the signals and I/O and then select them to see their waveform. Now looking at the first 2 waveforms, they represent the input of the systems that I decided in the testbench, first is the clock monitoring the design, and second emulates an action on the button. The third waveform is the answer of the output, here the led, and as expected the state change after an action on the button. Finally, other waveforms represent the RAM, and I can see that every time my led changes, a value is stored in my RAM.

Now that the design is validated in simulation it’s time to program a test on the board. Before that I set up the environment. To program the board, I use a JTAG connection. JTAG is a standard interface to program FPGA’ logic blocs and interconnexions. It also allows to access the internal registers and so to allows debugging features. In my case I use the Altera USB Blaster, connected on one of the USB ports of my computer and on the JTAG port of the board. Now I have the source code and programming JTAG connection I go on the pin planner to choose my I/O. The clock is obtained with an oscillator on the board, and I can see on the reference manual frequencies available [[3]](#footnote-3) and choose the corresponding pin. By the same methods I choose my input: the button, and output: the led and associate them in the pin planner.

Table

Description automatically generatedText

Description automatically generated

Figure 4: Pin planner and pin assignment

Now I can open the programmer tool. I have to choose the right device and programming mode, here JTAG, and I can program the FPGA with the .sof file generated in Quartus. Once the board operational I test by pushing the button associated and I can see the led changing it state [[4]](#footnote-4) (from on to off).

### MATLAB

Now I need to handle the FPGA in the Loop and it process. Before starting the project idea of generic testbench I will follow the workflow provided by MathWorks on their example "HDL Implementation of PID Controller Using FPGA-in-the-Loop" 4 .

A Proportional Integral Derivative (PID) Controller is a control feedback loop. The controller continuously calculates the error, difference, between a desired point and the real measure. According to the difference calculated it applies a correction to adjust the control output. In the MATLAB example, the PID controller is used to control the position.

The first step in this process is to ensure that all necessary components are in place. This includes the installation of a FPGA design software, which in this case is Quartus Prime Lite described above. The use of a board that is supported by the software and has a gigabit ethernet port, the Cyclone V GX respects these needs. And then a license for MATLAB: HDL tools coder and verifier which I obtained through a virtual machine provided by the school.

To run a FIL the current MATLAB session needs to access the FPGA design software, so I change the directory files until the path of the “quartus.exe” on my computer and run the command:

hdlsetuptoolpath('ToolName','Altera Quartus II','ToolPath','C:\Intel\quartus\bin64\quartus.exe')

Next, I setup the environment, the board is programmed with a JTAG connection, as seen before in the “Quartus” section but it communicates with the computer with the Gigabit Ethernet connection. The FIL can be done fully by JTAG but with a slower transfer rate and smaller amount of data. To setup the host computer-board connection I choose my host address. To do that I go to the control panel and choose for the project 192.168.0.1 because it is not already used.

Now I want to generate the Simulink block that will be send in a schematic and bear the VHDL files. So, I open the filWizard, and complete the fields depending on the needs. On my case what’s different from the example is the board used. I use the board manager to get my board with all its fields filled but it also can be done manually. Be careful, on my project after those steps I had problems, the board wasn’t connected well, to resolve that I went to “Board Manager”, selected my board, and click on “edit” “Interface” and in “advanced option” I ticked to generate MDIO module. According to MATLAB documentation this box needs to be checked when the connection is about to be done with RGMII mode with ethernet PHY device Marvell88E1111 which is the case for the cyclone V GX. After the board the second thing different is for the ethernet configuration, for the board IP address, since I choose for the host 192.168.0.1 so the IP address has to be a subnet of 192.168.0.x.

Then the pieces of information asked by the filWizard depends on the VHDL source files such as the top level, the I/O and their types and the type of the outputs data on Simulink. Once build, many things are generated:

* A FIL block on a Simulink page. Tis block contains the .sof and programs the board, then send and receive the data.
* A command window that performs synthesis, place-and-route, timing analysis and programming file generation. To do that a whole new project is generated and when have access to it to change it as wanted. The new project contains the sources files and the generated files containing for example the clock manager, Ethernet configuration or MDIO seen above.

I can now fetch the Simulink schematic example to evaluate the PID monitor. So, the simulation is performed by Simulink, but the PID is inside the FPGA that process the calculation.

Graphical user interface, chart

Description automatically generated

Figure 5: Results of the PID control performs by FIL

Looking at the results I can see that the actual position is close to the desired one, we have some exceeding and reel angles. To add more accurate value between desired and actual I can modify the values of the PID. But here the objective was to follow the FIL Workflow so now I can go on with the project.

## Adapter design and test

Now that the tools are basically understood. The goal is to create a kind of wrapper between the IP to be tested and the whole MATLAB tool communicating with the board as an adapter. The adapter will be designed in VHDL with Quartus environment, and it will be deployed on the board (and not Simulink from MATLAB) as the IP to be tested. It has two main goals:

- Adapt thy raw data on an Avalon Streaming Interface (AST) and then adapt the AST data to a message to the simulator.

- Manage the different timing domains.

For the project I start by designing the adapter and evaluate it with no IP inside. So, the IN adapter and OUT adapter are linked together and first test expect to send signals and see it at the output with no alteration.

Diagram

Description automatically generated

Figure 6: Adapter with no IP inside

### Raw design on Quartus

First, I’m working on AST bus. I have to choose which signals are needed. The idea is to start with basics. So, I fetch the AST documentation and look at the possible signals to decide which one are necessary.

Signals or I/O can be “source” or “sink.” Data transfer is started by the source block, by driving the data and valid signal. The sink block will then read the data and valid signal and indicate that data has been read by driving the ready signal. It comes the necessary signals:

* ast\_source\_valid ast\_sink\_valid -
* ast\_source\_data ast\_sink\_data -
* ast\_source\_ready ast\_sink\_ready -

Also, I said that before that the adapter must manage the time domain. One specification is to run the IP we want to test at 125 MHz But the Simulator on MATLAB isn’t working this fast, or if I want to force it will be exceedingly long and not relevant cause I won’t need such precision of signal while the goal is to see in real time.

Diagram

Description automatically generated

Figure 7: architecture of the Adapter

The adapter consists of two main components: the IN and the OUT. Each of these components has three blocks. The IP to be evaluated will be inserted between these two components.

The IN-adapter first block is the “IN-storing”, this block receives the data from MATLAB and a clock. Also, the block is linked to generated files to MATLAB, inside those files is an “enable” signal telling when a new data is sent from MATLAB. This block is a single synchronous process fetching data from MATLAB when a new ne is available and sending it to the next block.

The next block is a First-In-First-Out (FIFO) memory, a type of buffer memory that stores data in a first-in, first-out order. First-in first-out order means that the first data that has arrived will be the first to leave the memory. FIFO memory stores data temporarily as it is being transferred between different clock domains. The FIFO has two clocks for the write and read part. The write part is connected to the MATLAB wrapper and the read part is connected to the IP so at 125MHz clock.

The IN-adapter last block is the “IN-convert”, linked to the read part of the FIFO. It goal is to communicate with the IP, so it has to furnished the IP inputs: ast\_source\_valid and ast\_source\_data depending on the state of the FIFO and the ast\_source\_ready. In a nutshell; in a synchronous Finish State Machine (FSM), when a data is available at the FIFO, the adapter is valid and the destination is ready and only when these three conditions are respected so the ast\_source\_data can take the FIFO value and so transfer the data.

The OUT-adapter first block is “OUT-storing”. It is supposed to be connected right after the IP to be tested. The block’s goal is to catch the IP results and store it. In a nutshell; in a synchronous FSM, when the data of the IP is valid, the adapter is ready and the next FIFO can store a data and only with these conditions then the ast\_sink\_data is transferred in the FIFO.

The next block is a FIFO memory, it has the same role as the previous FIFO, to be a buffer between the two-time domains.

The last block is the “OUT-stream”, it is situated between the FIFO containing results and the wrapper from MATLAB. Once the Simulator can take a new value (with “enable” signal) and if there is a data to read in the FIFO then the FIFO is read, and data result send to the simulator.

As said previously, first thing is to check that the adapter works and don’t lose any value. So, I create a testbench, connect the adapter IN to the OUT and expect to see the same output as input. And in the Quartus simulation I will be able to check on AST signals.

Graphical user interface

Description automatically generated

Figure 8: ModelSim observation of the outputs

A screenshot of a computer

Description automatically generated On the figure 8 the circuit under analysis has 4 inputs: the clock, enable, reset and the input data. The clock is operating at 125MHz, reset is stuck at 0. Enable is supposed to represent a future MATLAB signal, meaning that new data can be read and the input data. The input data in this example is just a visual pattern. As expected, the output of the circuit is identical to the input, with a delay that corresponds to the clock cycles required for FSM, transfer data and later the computation. In other words this is the response time.

Figure 9: ModelSim observation of AST signals

On the figure 9 the circuit under analysis has as previously the clock at 125 MHz, reset stuck at 0. Now we want the data to change at a rising edge from the clock when all three conditions are met: valid data is present (ast\_valid), the destination is ready to receive (ast\_ready), and the enable input is set to 1 indicating new data is available (enable). We can see on the figure with yellow markers that the data change when all conditions are respected.

The design is now validated, and then we want to deploy it and the board with the FIL method.

### Design in FIL

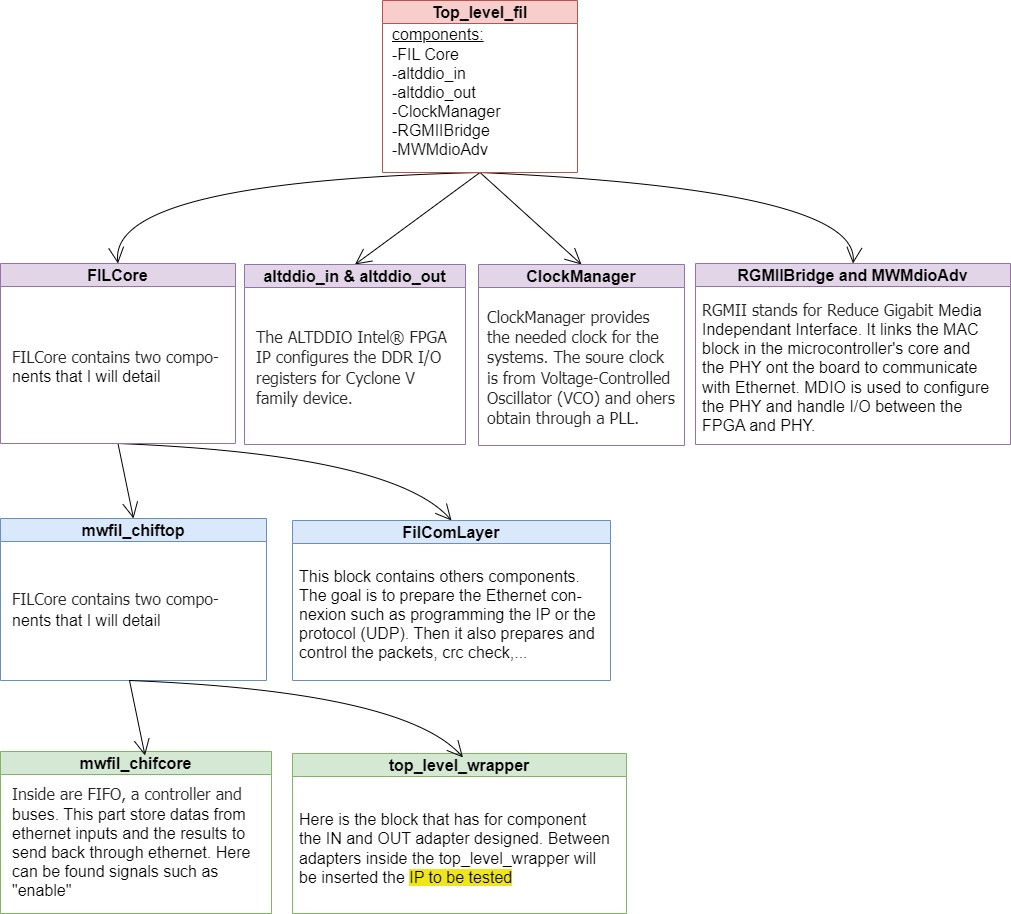
Now I open MATLAB and the filWizard, allowing to generated files for FIL. What I want to do is generate the block on Simulink and the VHDL project on Quartus. Later, on Simulink, I will simply send an input data signal. The clock, reset and enable are manage inside the project. The clock and reset are automatically managed and so don’t appear in the Simulink schematic. To manage the “enable” I comment the line of the top level with the input “enable”. This input will be connected to generated files later, so it is pending.

I follow the steps seen before where I select the source files, outputs, and top level. The format used is:

* input: fixedpoint(1,32,16); meaning a signed format with 32 bit width and 16 bits after the coma.
* output: fixedpoint (1,50,27); meaning a signed format with 50 bit width and 37 bits after the coma.

Once the FilWizard has finish we can open the FPGA project and the files generated. The idea is to understand these files, so we can uncomment the “enable” input and connect properly.

Figure 10: architecture of the generated project



The graph represents the architecture of the generated project. Each color on blocks represents a different layer of the architecture. Only majors IPs are described in a few words to understand their role in the whole project. The progression moves downward until the adapters, designed and test at the section above, which also is the location of the future IP to be tested (in yellow).

The fil\_top\_level, at the first layer, connects the different IP but has no process. At the second layer are IP allowing the communication between the board and the simulator and the clocks manager. The third layer, leading to the design, also includes the protocol and information necessary to configure the Ethernet. The fourth layer has a kind of controller of the I/O and a wrapper, wrapper containing at the fifth layer our design.

Diagram

Description automatically generatedNow that I changed the FPGA project, to fit specifications, I compile it to have the new binary file (.sof). This file will be used to program the board through the FIL block on Simulink. Before that I add on the schematic page some blocks to test and validate the project.

Figure 11: Simulink schematic to test FIL block.

To validate it, first step is to check that if I do not put any IP to test between adapters, I see the same inputs and outputs. In the figure I put 3 different kinds of inputs : a constant, a sinusoids and an additions of sinusoids. The input of the FIL block cannot support continuous data, so the inputs are sampled, and I choose the period at 50 microseconds for a 10 seconds simulation. I add converter to fit the input of the bloc and one after the FIL block to have double. At last, I use a scope with two ports, one for the input signal and one for the output of the block to compare them.

The Simulink schematic is ready so I setup the environment. I connect the board with both connection JTAG and Ethernet. The FPGA project is flashed on the board with the JTAG.

At this point I had an issue: the output was stuck at 0. After some tests with Signal Tap I couldn’t see any evolution of the signal “enable”, signal supposed to start the transfer through adapters at rising edge. The issue is explained because the clock injected in my IP is no longer 125Mz but replaced in the generated files. By default, it’s the “not(dut\_enable)” which is assign to our design as a clock and “dut\_enable” is the one I assigned to my “enable input”. Because both signals are opposite I never start the process so I have no results. This signal means that MATLAB has send a new value and the value is available to read. I want this signal to be to my “enable” input and I want my IP to work at 125MHz. The top\_level\_wrapper is my adapter IP parent [[5]](#footnote-5), in the file I add the assignments. “dut\_enable” to “enable”, that was before assign to the clock. For the clock, first I take a look to the timing reports and clocks summary. There I can identify the frequencies out of the Clock Manager and confirm that the clock used in the top\_level\_wrapper is at 125MHz, finally I can assign it to the clock of my top\_level.

Running first tests, I send a constant value but see some glitches on the scope of Simulink. I can check on the ethernet traffic by Wireshark. Wireshark is an open-source network protocol analyzer. It allows to capture network packet, their protocol, data transfer, and the order of the packet transfer. To understand if glitches were from MATLAB or the board I capture the Wireshark packets from the board, and I can see that the constant value is mostly here but some random data too. I conclude that the parasites were from the board. The data has no logical reasons and seems to be random because for a same input send the result is different. A solution used is to reduce the ethernet speed transfer, notably through the clock of the IP MWMdio. To simplify I do not create any new clock but I use one of the Clock Manager. There are 125MHz and 25MHz so I take the one at 25MHz because I want to reduce.

Graphical user interface

Description automatically generated

Figure 12:results of FIL with adapter and no IP

This example shows when a sinusoid is injected has input and that I have the same one at the outputs. I estimate that it is the same because it is the same frequency and amplitude. I also run it for constants input and combination of different signals.

The scope allows to validate visually but also to validate it fully I realized a subtraction between input and output and I look at the workspace the results.I want a zero or close to zero. For the simulation on figure 12 I have an average error around 0.22. This error is mainly constant with a tiny variation and. In my opinion, it is explained by a shift between the input and output. The shift was more visual in figure 8: showing on quartus and on a time scale more precise ( hundreds of nanoseconds) but on the scope the time scale is about seconds so we can’t see the shift.

I consider now that the adapter isn’t modifying or deleting datas and that it manage the AST bus. Next part is an example on how to use it with an IP.

### Test the design with a filter

In the previous part we’ve seen that the adapters were functional without any IP inside. Meaning that at least no data are lost or changed. The idea of this section is to try and test it using an example of an IP. The IP I will try is a filter.

A filter is a device that removes unwanted components or features. I will try on a Finite Impulse Response (FIR), a digital filter characterized by coefficient. This example allows to implement the assets of the FIL, a real time around signal processing. With this filter I want to work with a sampling rate of 4000Hz, the IP working in 125MHz on the board. I can look the timing response over a second, find out the coefficients and frequency response all of it with MATLAB tools.

First of all, I open the FPGA project to add the IP in the” top\_level”. New signals are created: three for the sources and three for the sink, representing data, valid and ready detailed before. The IP also has signal for “ast\_error” that I stuck at 0.

Careful, some reset signals can be either active on high state or active on low state. Meaning that all registers, signals or state returns to initial values when this signal is high (at 1) or low (at 0). Precisely in the project it a reset active at high state while the FIR is active at low. To resolve this, a new signal, "reset\_n = not(reset)", is created and assigned to the IP.

Diagram

Description automatically generatedThe project is ready and after compilation I have the binary file that will be used in the FIL block and then program the board. Now I create the schematic on Simulink.

Figure 13: Simulink schematic for the FIR

First thing is to look at the response to a Dirac. Dirac means that it is always to 0 but not at 1 single moment. For digital studies it is a Dirac to one. Then the converter is used to get the right format (fixedpoint(1,32,16)) inside the FIL block. The FIL block communicates with the board and it results are sampled at 4000Hz and converted. Finally the scope allows to see the input and output with timing scale. But now I don’t only want to see the timing response I want the coefficients. To have them clear I add in my scope a parameter variable (variable will take the value received), this variable can be used in the workspace and so I will be able to get values.

Chart, histogram

Description automatically generatedI do run the simulation, open the workspace and the variable set. I see the response of the Dirac, select the values and plot it:

Figure 14: Coefficients of a FIR

The impulse response is said finite if the number of non-zero samples is finite. On the graph I can see the 37th coefficients defining the finite response and more generally the filter. Also, the response looks like a cardinal sin, so I expect a low pass filter. Now I copy coefficients into the Filter Wizard. FilterWizard is MATLAB’s tool to analyze digital filters. With the wizard I can see the frequency response and the phase response.

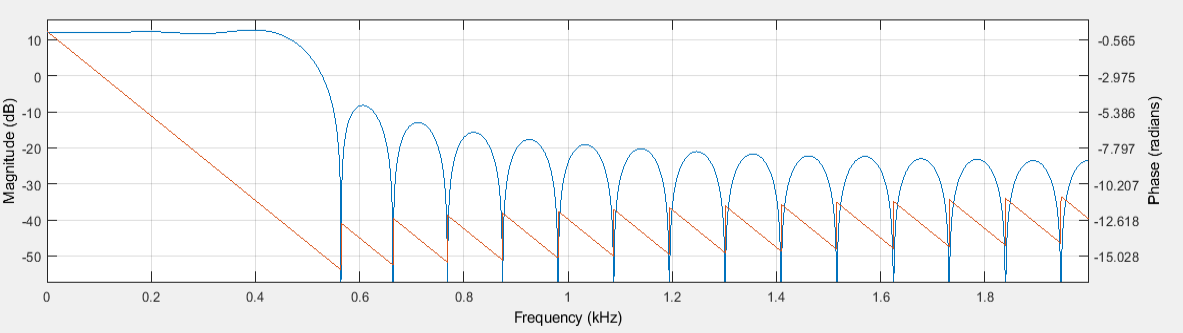
First, I open the Wizard and click on the section “import filter from workspace” at the left border. At the “filter structure” I choose “Direct Form FIR”, in sampling Frequency I choose “Hz” for unit and “4000” for Fs. Finally, I can add the coefficient fetched before and import filter:

Figure 15: Magnitude and Phase response

The blue print is the magnitude response. We can see the low filter as expecting cutting around 500Hz. It means that it signals with frequency higher than 500Hz are going to be attenuated.

The orange represents the phase, and we can see that it’s linear in the bandwidth. A linear phase means the phase shift at the input is linear with the frequency.

# Conclusion

## Feedback on results

Before concluding on results, I will set up a reminder on the objectives.

This project was about to be able to test VHDL IPs more efficiently than classicals ways: simulation or prototyping. The tool chosen was FPGA-In-the-Loop. FIL is a complement of behavior simulatior such as ModelSim from Quartus. ModelSim is done for simulation at one cycle close, to control in detail the behavior or timing. On the contrary FIL allows to use a large range of libraries or mathematics tools through MATLAB features therefore to work on applicative test, in real time. Another specification of the project is to have a generic benchmark, not to have to generate a new for each IP. To manage that I consider that the IP to test are on Avalon Streaming interface (AST).

At the end of the project I have functional adapters to welcome an IP with AST I/O and a schematic on MATLAB Simulink to communicate in real time with the board and so test and validate it. In Quartus I could have look at the intern signals of AST and on MATLAB look at the outputs. The addition of a filter is one example of the interest to use Simulink graphical tool notably for signal processing. I send a Dirac using an existing block and can store or draw response.

But I can notice two main limitations on the actual design:

-The AST adapters are limited. Many other signals exist on AST but only necessary ones are here. Some IP can be tested however, if there are more signals, just stuck them at zero or one depending to their role. But in some cases, IP can’t be tested, for example the ones with channels can’t be inserted and tested in the actual design.

-The FIL block in Simulink output are stuck. Now if I want to test an IP with more inputs or bigger one I will have to re-generate and repeat to step to link the enable to the wrapper and change the clock.

## Notions learned and competences

On this project many notions were addressed. I will here review them in the order I met them.

First one, I completed my ability to work on new board and development tool. I work on Intel environment (board Cyclone V and Quartus). Generally for the board it was about the frequencies of intern oscillator, I/O and pins and methods to flash and communicated with the board and for the development tool Quartus I’ve learn the basics like the different sections available on the window, how to program, debug behavior with ModelSim or Signal Tap or debug timing with the timing analyser. Also, to work on JTAG chain and programmer tool.

Afterwards, I’ve worked on the FPGA-In-the-Loop. Precisely I’ve worked on the needs of the tool such as communication, flow, clock. I had to understand how it was working to then use it and understand how it would be linked in with an FPGA project. A project and VHDL files are generated so the idea was to dig in it to understand what was in a direct link with future adapters, the clock and I/O.

Then was the design of adapters. To design it I discovered and used Avalon from Intel. I’ve look at documentations to have a context of this protocol, when and why it is used and examples to understand what signals were necessary and how to use them correctly.

To link the board and the Simulink schematic I used ethernet. I work on the set up of the environment such as the host, work on the UDP protocol specification and the control of traffic with Wireshark. Additionally, many files are generated to configure the ethernet on the board. It was an introduction to RGMII protocol.

In addition, I’ve tried it with a filter application. This lead to the study of how to characterize filter, look at the response to Dirac, timing or frequency, estimate the outputs and understand what I have for results. To do that I used the MATLAB and Simulink libraries and mathematics tools.

Finally this project was a deeper approach of the engineer work. I work on the process consisting in meeting to see the progression, weekly reports and deadlines.

## What next

This project was an introduction for notions and to prepare the tool FIL. Next will be an internship with the partner company GE. This internship consists in developing a FFT (Fast Fourier Transform) on FPGA on signals from electrical grids. The goal is to use the FFT to analyze the spectral responses of these signals with the FIL tool that was implemented during the project.

In other words, it’s about to replace the FIR seen above by FFT. FFT will be done in Quartus (VHDL) or with the Intel DSP builder library, a library used to performs FFT on FPGA through MATLAB.

# Acronyms

A

AST: Avalon Streaming · 4, 5, 6, 10, 11, 12, 16, 19

F

FFT: Fast Fourier Transform · 20

FIFO: First In First Out · 11, 12

FIL: FPGA In the Loop · 2, 4, 5, 7, 8, 9, 10, 13, 15, 17, 19

FIR: Finite Impulse Response (filter) · 17, 18

FPGA: Field Programmable Gate Array · 1, 2, 3, 4, 5, 6, 7, 8, 9, 13, 15, 17, 19

FSM: Finish State Machine · 12

I

I/O: Inputs / outputs · 6, 7, 8, 9, 11, 14, 15, 19

IP: Intellectual Property · 3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 19

M

MDIO: Management Data Input/Output · 9

P

PFE: Projet Fin d'Etude · 3

PID: Proportional Integral Derivative · 8, 9, 10

PLL: Phase Locked Loop · 7, 8

R

R&D: Research and Development · 4

RAM: Random Acces Memory · 7

RGMII: Reduce Gigabit Media Independant Interface · 9, 20

S

SoC: System on Chip · 4

# Useful links

* Code used

*GitHub repository of the End of Studies’ project | FPGA\_In\_the\_Loop\_cycloneV. Available at* [*https://github.com/camilleherrmann/FPGA\_In\_the\_Loop\_cycloneV/tree/master/FIL\_application*](https://github.com/camilleherrmann/FPGA_In_the_Loop_cycloneV/tree/master/FIL_application)

* Board documentation

*Altera Cyclone V GX Development Kit (Discontinued) Circuit Note | Analog Devices*te). Available at: <https://www.analog.com/en/design-center/reference-designs/circuits-from-the-lab/altera-cyclone-v-gx-development-kit-discontinued.html#rd-overview>

*Cyclone® V GX FPGA Development Board Reference Manual* (no date) *Intel*. Available at: <https://www.intel.com/content/www/us/en/content-details/654409/cyclone-v-gx-fpga-development-board-reference-manual.html>

* FIL documentation

*FPGA-in-the-Loop Simulation - MATLAB & Simulink - MathWorks Nordic* (no date). Available at: <https://se.mathworks.com/help/supportpkg/xilinxfpgaboards/fpga-in-the-loop-simulation.html>

*Verify HDL Implementation of PID Controller Using FPGA-in-the-Loop - MATLAB & Simulink - MathWorks France* (no date). Available at: <https://fr.mathworks.com/help/hdlverifier/ug/verify-hdl-implementation-of-pid-controller-using-fpga-in-the-loop.html>

* Avalon documentation

*mnl\_avalon\_spec-18-1-683091-705418.pdf | Powered by Box* (no date). Available at: <https://app.box.com/file/1049224181509>

*Introduction to the Avalon® Interface Specifications* (no date) *Intel*. Available at: <https://www.intel.com/content/www/us/en/docs/programmable/683091/20-1/introduction-to-the-interface-specifications.html>

* Intel generated files

*Double Data Rate I/O (ALTDDIO\_IN, ALTDDIO\_OUT, and ALTDDIO\_BIDIR)...* (no date) *Intel*. Available at: <https://www.intel.com/content/www/us/en/docs/programmable/683148/17-0/double-data-rate-i-o-altddio-in-altddio-54289.html>

*MDIO* (no date) *Intel*. Available at: <https://www.intel.com/content/www/us/en/docs/programmable/683117/20-4/mdio.html>

1. The company and its role will be described in the second part of the introduction: “Company and project idea.” [↑](#footnote-ref-1)
2. The code use is available on the GitHub repository of this project in the folder “FIL\_get\_started\_tuto” and “test\_btn\_memory\_for\_Quartus”. Link of the GitHub repository can be found at the end of the report in the part “useful links” [↑](#footnote-ref-2)
3. If the frequency desired isn’t available we still can use a PLL. [↑](#footnote-ref-3)
4. A video of the result and detailed steps with pictures are available the GitHub repository of this project in the folder “FIL\_get\_started\_tuto. Link of the GitHub repository can be found at the end of the report in the part “useful links”. [↑](#footnote-ref-4)
5. The IP parent is at one layer before and as for component the one we talk about. Connecting is I/O to the rest of the system. [↑](#footnote-ref-5)