University of Sevilla

Architecture and Technology of Computers Dept.

Robotics and Technology of Computers Lab.

OpenNAS tutorial

Authors:
Daniel Gutierrez
Juan Pedro Dominguez
Angel Jimenez
Alejandro Linares

November 28, 2018

Abstract

OpenNAS is a software tool to customize, create, and generate a Neuromorphic Auditory Sensor (NAS) model ready to be loaded within the FPGA-based AER-Node board. It is the result of many research years in the field of neuromorphic engineering, in which our research group has enough experience. As its name indicates, this spike-based auditory sensor tries to mimic the human ear and the way in which it works. OpenNAS can be divided in three main blocks: the input, where the sound is obtained, digitized and converted into spikes; the spike-based audio processing step, where the input spikes are decomposed into frequency channels, and, finally, the output stage, where the output interface through which the spikes are going to be sent is selected. Since there is not a standard "cochlea configuration", OpenNAS tool offers the possibility to set a large set of parameters with the aim of providing the best approach for the required spike-based audio applications.

1 Introduction

"This paper presents a new architecture, design flow and FPGA implementation analysis of a neuromorphic binaural auditory sensor, designed completely in the spike-domain. Unlike digital cochleae that decompose audio signals using classical Digital Signal Processing (DSP) techniques, the model presented in this paper processes information directly encoded as spikes using Pulse Frequency Modulation (PFM) and provides a set of frequency-decomposed audio information using an Address-Event Representation (AER) interface. In this case, a systematic approach to design led to a generic process for building, tuning and implementing audio frequency decomposers with different features, facilitating synthesis with custom features. This allows researchers to implement their own parameterized neuromorphic auditory systems in a low cost FPGA in order to study the audio processing and learning activity that takes place in the brain. In this paper we present a 64 channel binaural neuromorphic auditory system implemented in a Virtex-5 FPGA using a commercial development board. The system was excited with a diverse set of audio signals in order to analyze its response and characterize its features. The neuromorphic auditory system response times and frequencies are reported. The experimental results of the proposed system implementation with 64 channels

stereo are: frequency range between 9.6Hz to 14.6KHz (adjustable), maximum output event rate of 2.19Mevents/sec, power consumption of 29.7mW, slices requirements of 11,141 and system clock frequency of 27MHz."

This paragraph belongs to the NAS [2] original paper (its abstract). At the beginning, the idea was to design, develop and test spike-base processing blocks, which we called building blocks, to offer the community a way to develop more complex spike-based systems. We succeed in the development of spike-based motor controllers and spike-based filters.

Spike-based motor control blocks implement complex operations. However, in the spike domain, those operations are easier to solve than using the classical methods. One example of this simplification could be... Basic operations, like additions and subtractions were also implemented in the spike-domain. More information about how those operations work can be found in [3].

One of the most important spike-based block designed in this work was the band-pass filter, because it allowed to implement a filter bank, and then, to decompose a input signal into different frequency bands. It made it possible to work on our own Neuromorphic Auditory Sensor. The inspiration was taken from important authors as Lyon [4], among others, who developed bio-inspired analog cochlea model.

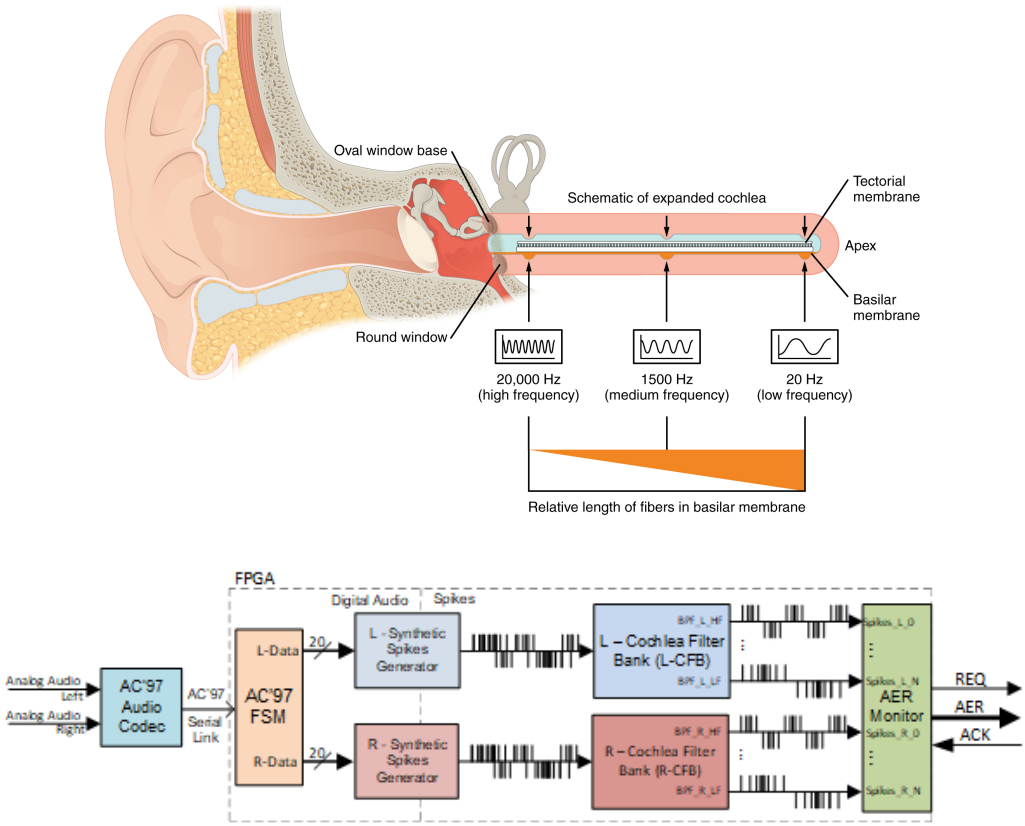

Figure 1: Top: biological model of human ear. Bottom: NAS architecture block diagram.

As a complex system, and based on the state of the art, the NAS is composed by several spike-based processing blocks, and it could be separated into three well differentiated and independent parts:

- Audio Input,
- Processing stage,
- Output interface.

And don't be afraid if the NAS doesn't work as you hope. You will have infinite tries to get the perfect one! :)

By using OpenNAS software, the user will be able to configure each of those three parts separately. The tool has been developed to easily configure and generate.....

2 OpenNAS overview

2.1 Requirements

- **SOFTWARE**: First of all, we need to check some software requirements to be sure that OpenNAS tool is going to work properly. All the software packages belong to Microsoft .NET Framework. Therefore, we will need a **Microsoft Windows OS** version running in the computer (Windows7 or Windows10). If you are not a Windows user (probably because you prefer a Linux distribution), you will need a Virtual Machine (VM) and a Windows installation. Furthermore, you should download the needed software. As IDE, **Visual Studio** was used since it provides an easy way to design Graphical User Interfaces (GUI) for Microsoft Windows OS. We will also need an specific framework, which includes features used in our software tool. More specific, we need to download and install both **.NET Framework 4.0** and **.NET Framework 4.5**.One of the best advantages that OpenNAS provides is the capacity of automatically generating the FPGA programming file (.bit extension) without either configure or create any project. This option is useful for people who are not familiarized with VHDL design. But, independently of what you need, you have to install **Xilinx ISE or Vivado** (depending of the board that you use). For the AER-Node board (device by default), you will need ISE. For the rest, Vivado for Xilinx FPGA chips (supported by OpenNAS); or Intel Quartus Prime Design Software for Altera FPGA chips (not supported yet).Finally, since we are going to use a specific manufacturer, you will also need a proprietary software to download the programming file into the FPGA board. **iMPACT** allows you either programming the FPGA using the RAM memory (volatile option) or the FLASH memory (non-volatile option). It is usually installed at the same time that ISE is being installed. But if not, you can go to the Xilinx Download Center website and download the software.

- **HARDWARE**: About the hardware requirements, you can use your own boards, but we can not offer any guaranties about the system operation. If you are interested on to use our neuromorphic hardware, please visit the COBER web site and check the neuromorphic hardware catalog.

IMPORTANT NOTES:

1. Versions are important: you must be sure that your installed software versions can support the OpenNAS project. Here there is a list which indicates the needed versions to run OpenNAS:
   - Microsoft Windows OS: Windows7, Windows8 or Windows10.
   - Visual Studio: Visual Studio 2015 or latest.
   - .NET Framework: you will need two: .NET Framework 4.0 and .NET Framework 4.5
   - Xilinx ISE Design Suite: ISE 14.7 **(It needs to support Spartan6 family chip)**
   - Xilinx Vivado: any version of Vivado.
   - Xilinx iMPACT: along with the ISE installation file.
2. More important things coming soon...

2.2 Design flow

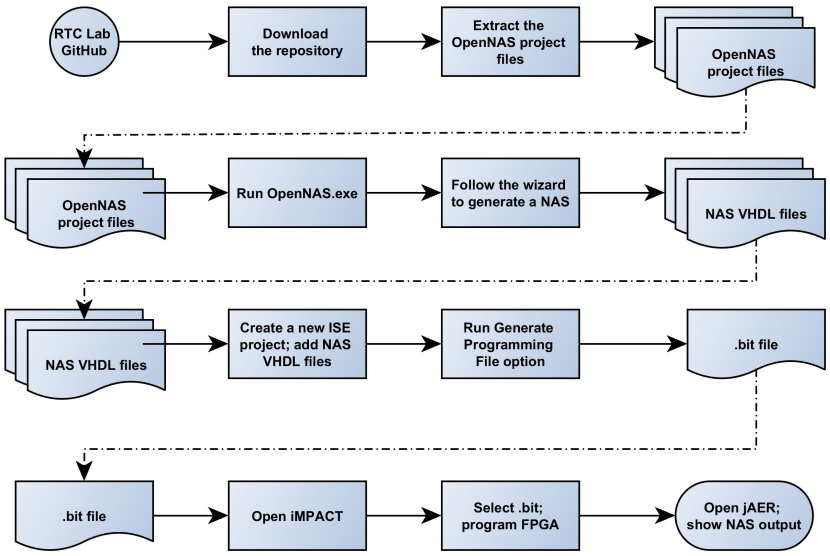

Figure 2: Complete design flow to work with NAS, starting from the GitHub repository to showing the NAS output by jAER.

2.3 OpenNAS configuration options

2.3.1 Commons

- NAS Chip
  1. AERNODE
  2. ZTEX
  3. SOCDOCK
  4. OTHER
- NAS Type
  1. MONO
  2. STEREO
- Number of Ch.
  - Minimun: 2; Maximun: infinite; Recommendable: values that are 2-pow, as 8, 16, 32, 64, 128 or 256.
- Clock Freq. (MHz)
  - Depend of your target board.

2.3.2 Input

- Select Audio Input:
  1. AC'97 Codec
     - Audio Input
       * LINE_IN
       * MIC_IN
     - Select Spikes Generator Settings
       * Spikes Gen bits
       * Spikes Gen Clock Div
       * KspikesGen (GSpk./s)
       * Max Spike Rate: (KSpk./s)

2. I2S Audio ADC
   - Select Spikes Generator Settings
     * Spikes Gen bits
     * Spikes Gen Clock Div
     * KspikesGen (KSpk./s)
     * Max Spike Rate: (KSpk./s)
3. PDM Mic
   - PDM Settings
     * System Clock (MHz)
     * PDM Clock Div
     * PDM Clock (MHz)
   - Anti-Offset SHPF Settings
     * Cutt-off Freq. (Hz)
   - Anti-Aliasing SLPF Settings
     * Cut-off Freq. (Hz)
     * Gain (dB)
4. I2S + PDM Mic
   - This option contains all the parameters listed before.

2.3.3 Processing

- Select NAS Architecture
  1. Cascade SLPF Architecture
  2. Parallel SLPF Architecture
  3. Parallel SBPF Architecture
- SLPF Filter: only when either Cascade SLPF Architecture or Parallel SLPF Architecture is selected.
  1. Order 2
- Number of Channels
- Start Freq. (Hz): according to the human ear, should be 20 Hz.

- Stop Freq. (Hz): according to the human ear, should be 22 KHz.
- SBPF Q Factor: only when Parallel SBPF Architecture is selected.
- SBPF Output Att. (dB)
- Target Mid Frequencies (Hz)
- SLPF Cut-off Frequencies (Hz)
- SBPF Q Factor: only when Parallel SBPF Architecture is selected.
- SBPF Attenuation (dB)

2.3.4 Output

- Select Spikes Output
    1. Spikes Distributed Monitor
        - Spikes Distributed Monitor
            * Spikes to monitor
            * Spikes FIFO Bits
            * AER FIFO Bits
            * Total Bits
    2. SpiNNaker-AER Interface v1
    3. SpiNNaker-AER Interface v2
    4. SpiNNaker-AER Interface v1 + Spikes Distributed Monitor
    5. USB Interface

3 OpenNAS wizard: how to use step by step

At this point, we are ready to start using the OpenNAS software tool. Maybe, you now can think "Did I not ready before to read about spike-based signal processing?" Well, for getting started you do not need to be a professional on that topic, but if your target is to implement complex systems, it would be good to know how the NAS works.

We will split the tutorial to use OpenNAS in three parts: first we need to download the project from the NAS GitHub web page. After that, we have

to either run the OpenNAS .exe file or open the OpenNAS VisualStudio project, in order to follow the OpenNAS wizard. And finally, to create an ISE project to generate the programming file needed to be loaded into the FPGA.

Let's do it!

3.1 OpenNAS GitHub repository

First, click on the OpenNAS GitHub website. If the link does not work, you can also open a web browser, and then copy there the OpenNAS GitHub website link: https://github.com/RTC-research-group/OpenNAS.

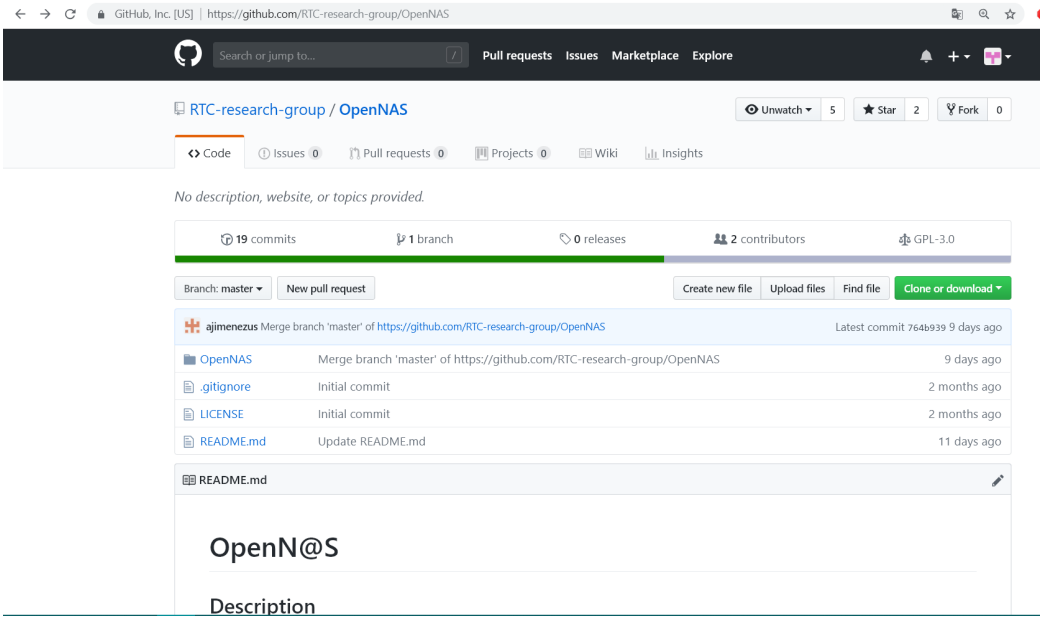

Figure 3: OpenNAS GitHub repository.

If that link does not work either, try to navigate to the RTC research group and look for the OpenNAS repository (https://github.com/RTC-research-group). There you can find a useful tool set to work with AER-based devices.

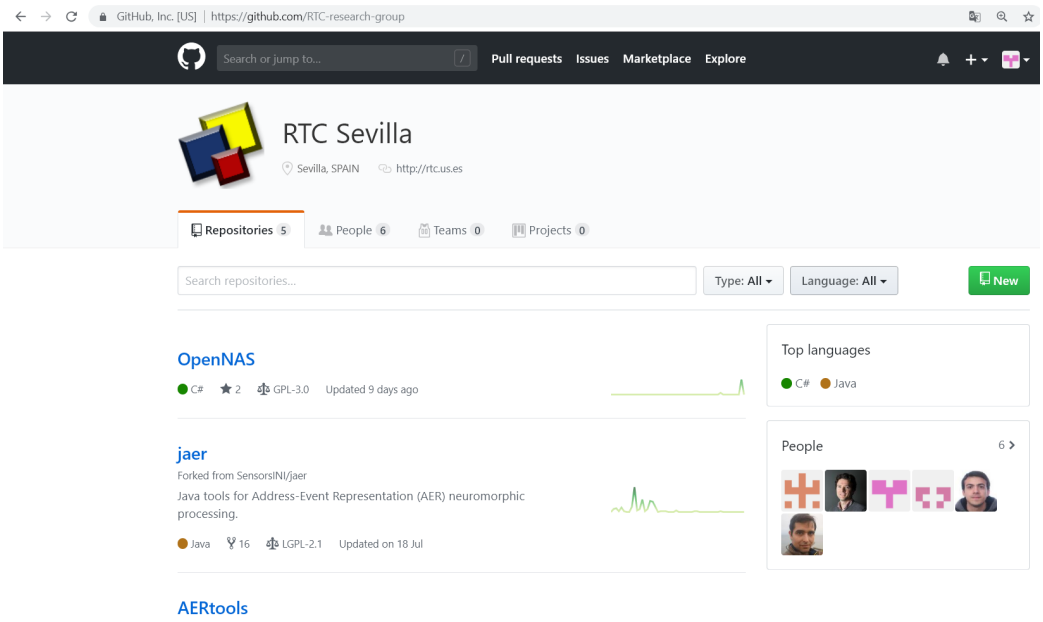

Figure 4: Robotics and Technology of Computers Lab. (RTC) GitHub page.

Once you are in the OpenNAS repository, there you can find a short wiki/tutorial about what is OpenNAS, how to getting started, how to use, how to contribute and, of course, how to cite our work. For us is very important the citations, so don't be shy. That wiki will be useful, but you can find more detailed information in this user manual.

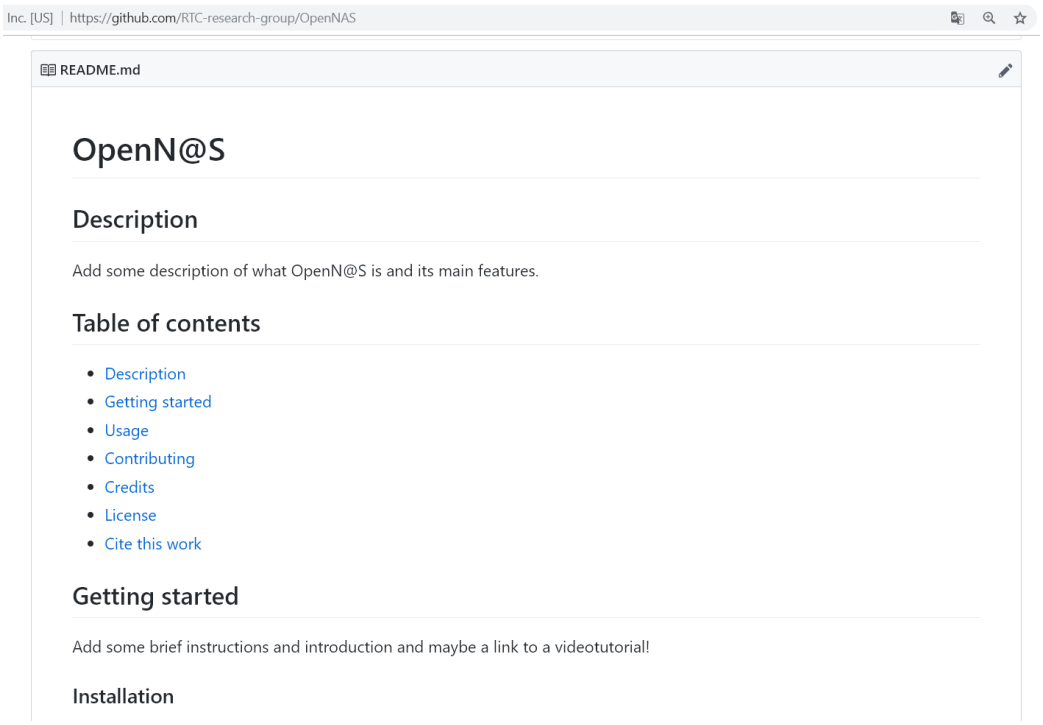

Figure 5: OpenNAS wiki overview.

The next step is either to download the project in your computer or to clone it. It depends on what do you prefer. And feel free if you want to contribute. You will be welcome!

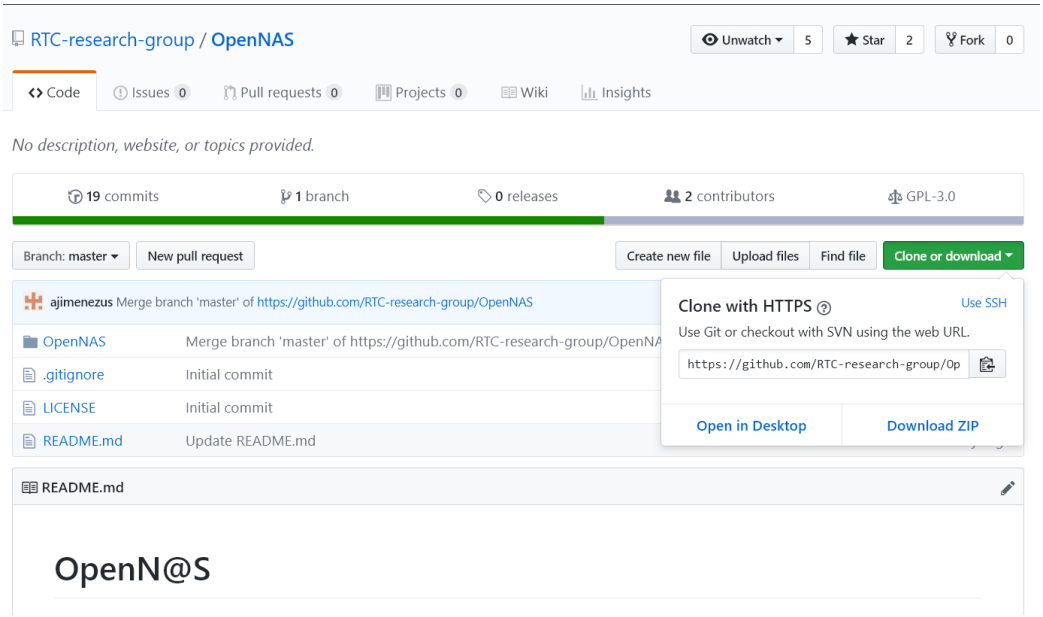

Figure 6: How to either clone or download the OpenNAS GitHub repository.

If you take a look within OpenNAS main folder, there are several subfolders which contain .VHD files (for FPGA) and .cs files (for C# application). The software hierarchy has been explained more detailed in section "Software architecture".

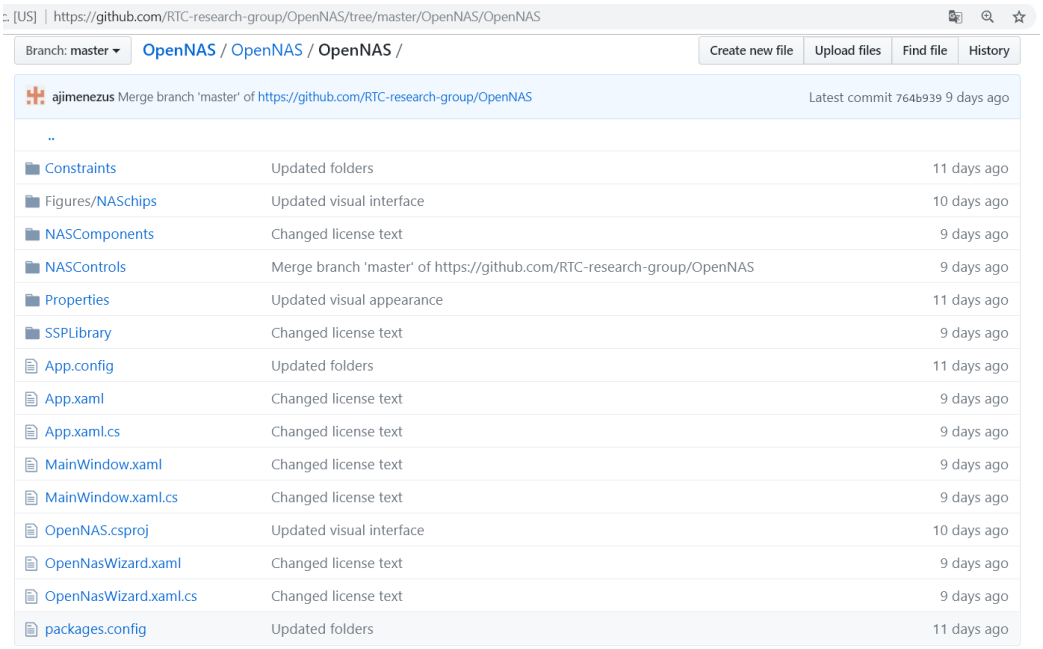

Figure 7: OpenNAS main folders.

Constraints folder is an important folder. It contains the files that you need to have well configured to use the NAS in your FPGA-based board. By default, OpenNAS includes three specific constraint files for three different boards: AER-Node board, SOC DOCK board, and ZTEX 2.13 board. If another board will be used, you must have to set your own constraint file.

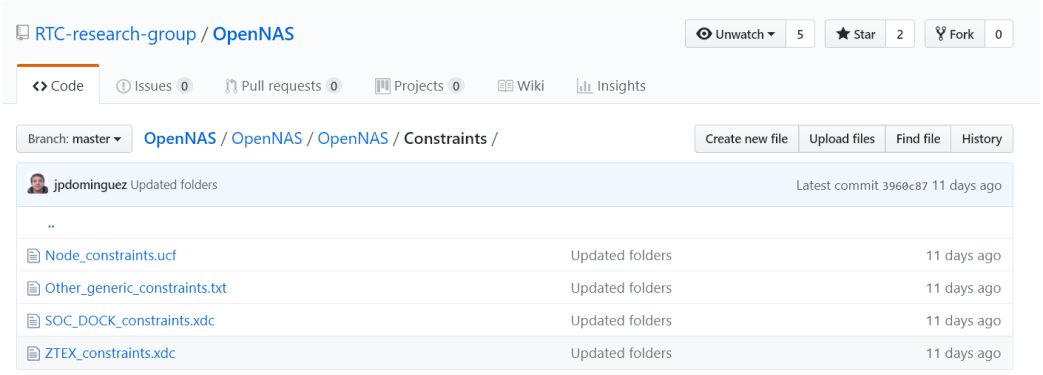

Figure 8: Constraints files available from OpenNAS.

Take care with that! This file is the main errors source.

Related with the VHDL files, the SSP (Spike-based Signal Processing) folder contains all of the source code of the NAS. You should not modify those files, otherwise NAS could not work properly. But you can check how the NAS has been implemented using HDL code.

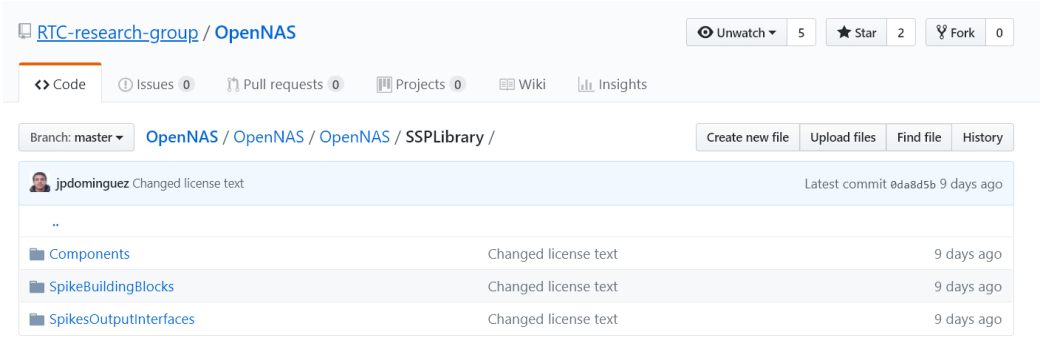

Figure 9: SSP folder contains all the NAS VHDL files.

Now, just a quick view of how those files look in the GitHub repository. This was showed clearly in the Software Architecture section.

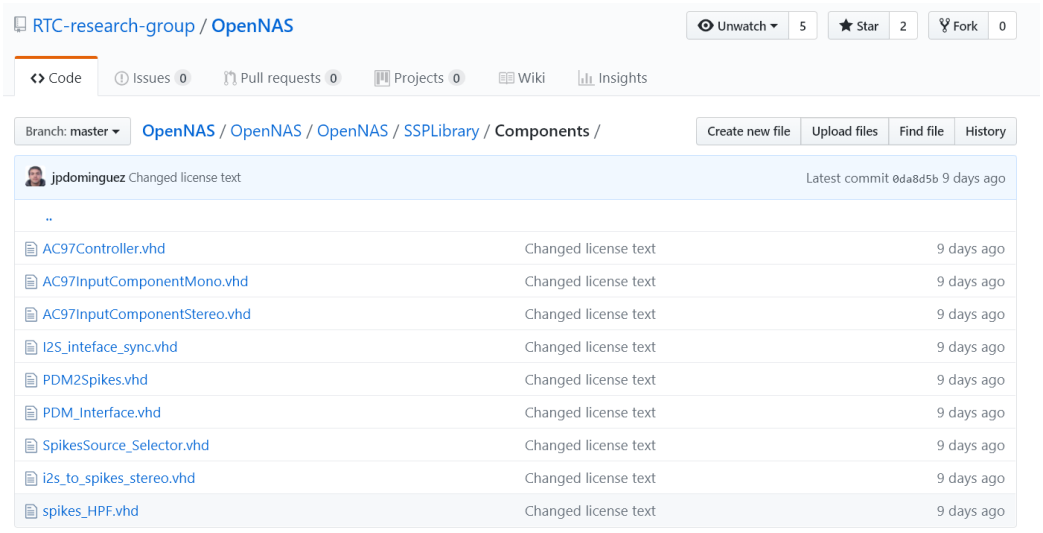

Figure 10: NAS input VHDL modules.

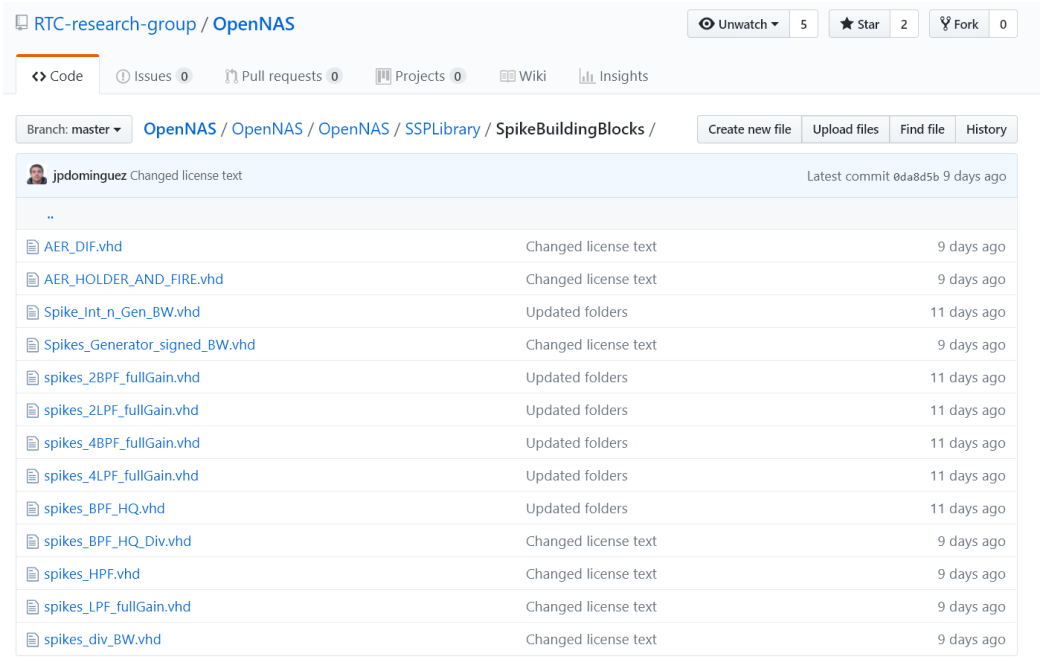

Figure 11: NAS spike-based signal processing VHDL modules.

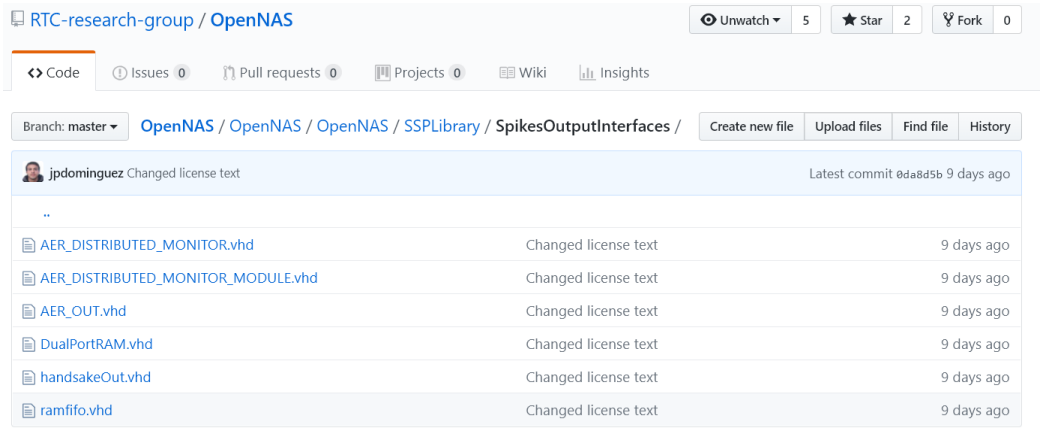

Figure 12: NAS output interfaces VHDL modules.

3.2 Running the VS application

After to either download or clone the repository, it is time to run the OpenNAS application. The first step is to unzip the project. You will have into the new folder something like this:

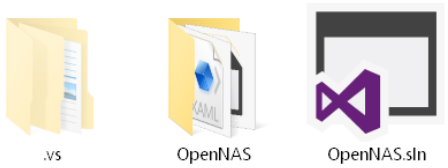

Figure 13: OpenNAS C# project files.

For running the software, you first need to open the OpenNAS solution (OpenNAS.sln), and then the VisualStudio environment will be launched. If it is the first time that you open the solution, an advertisement message should appear. Don't worry about that, it is just a warning message about malicious code.

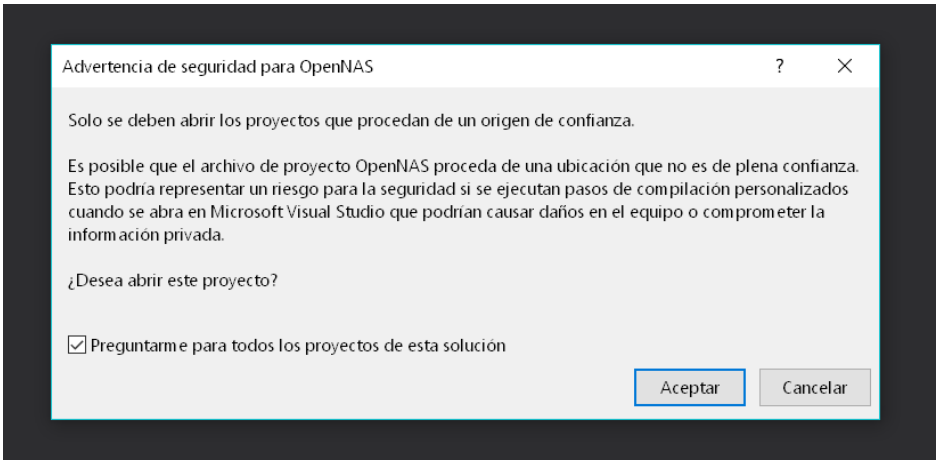

Figure 14: Advertisement launched by the VisualStudio software the first time you open the OpenNAS project.

Click on "Accept", and the main window project will appear. The user can change the appearance of the OpenNAS tool, but never should change the controls, because the application will not work.

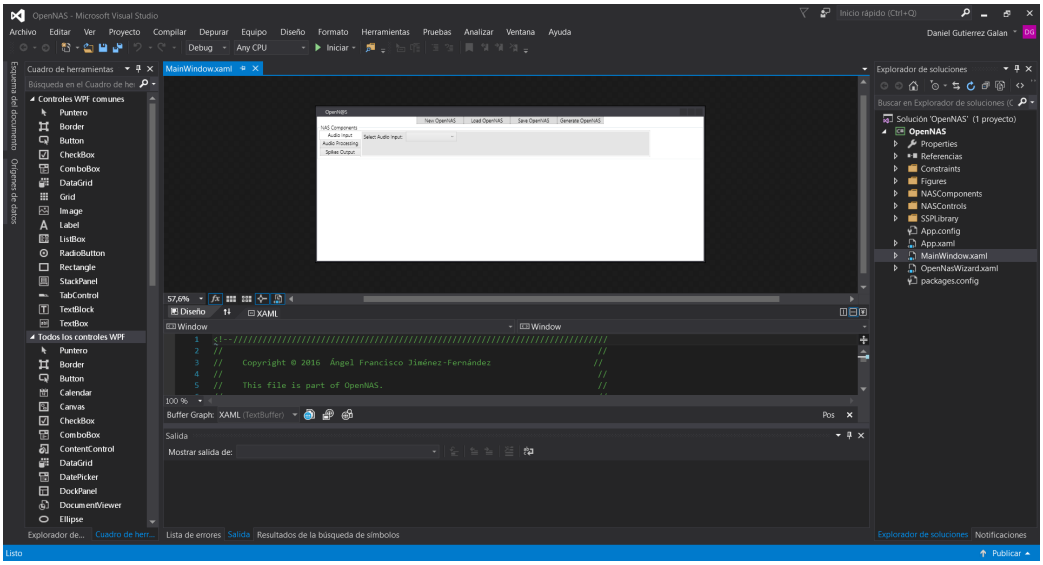

Figure 15: General view of the VisualStudio interface after open the OpenNAS project.

Now, the most difficult part: click on the "Start" button. A new window like the next one must appear...

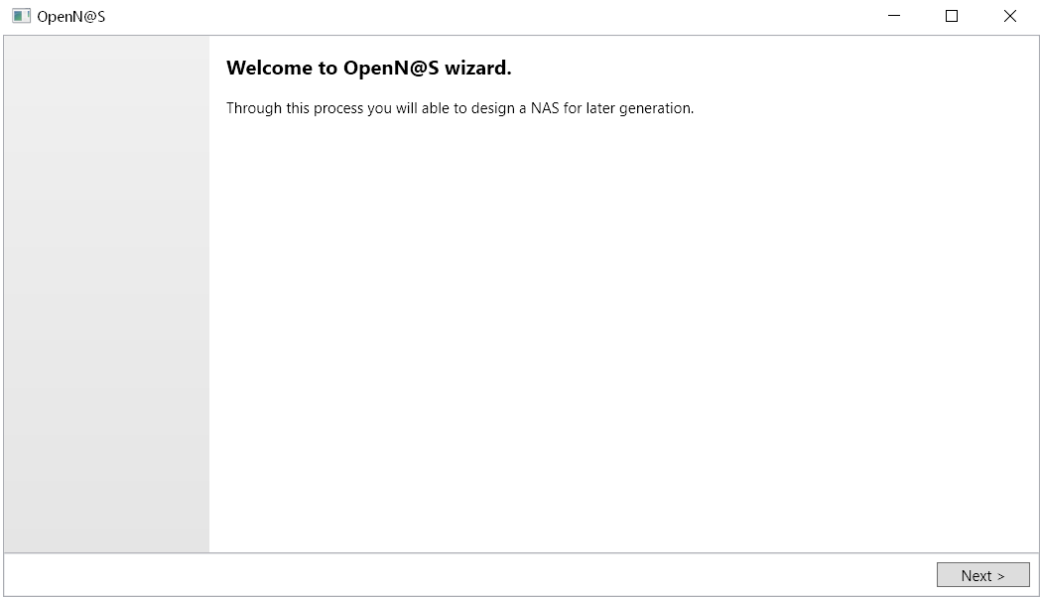

Figure 16: OpenNAS wizard main window.

CONGRATULATIONS, OpenNAS is now running! That is the main window of the wizard. It just has one button to go forward to the wizard. Click on "Next" button to watch the next step of the NAS generation wizard.

In the step 1, users can select the NAS common settings, as the FPGA-based board in which the NAS will be loaded. NAS type (mono or stereo) and the numbers of channels can be set too. Be careful with the last field, the clock frequency. It will be used to set the spike-based processing blocks parameters correctly. By default, each board set this field automatically according with the board clock (except if you choose "Others"). However, if you want use another clock configuration, you can change the clock value. **Make sure you also implement this change in the VHDL files and ISE project!**

Figure 17: First step of OpenNAS wizard. In this step, NAS common settings can be changed.

Here there is another option of board where NAS has been tested. If we check the clock frequency, its value has been set to 100 MHz.

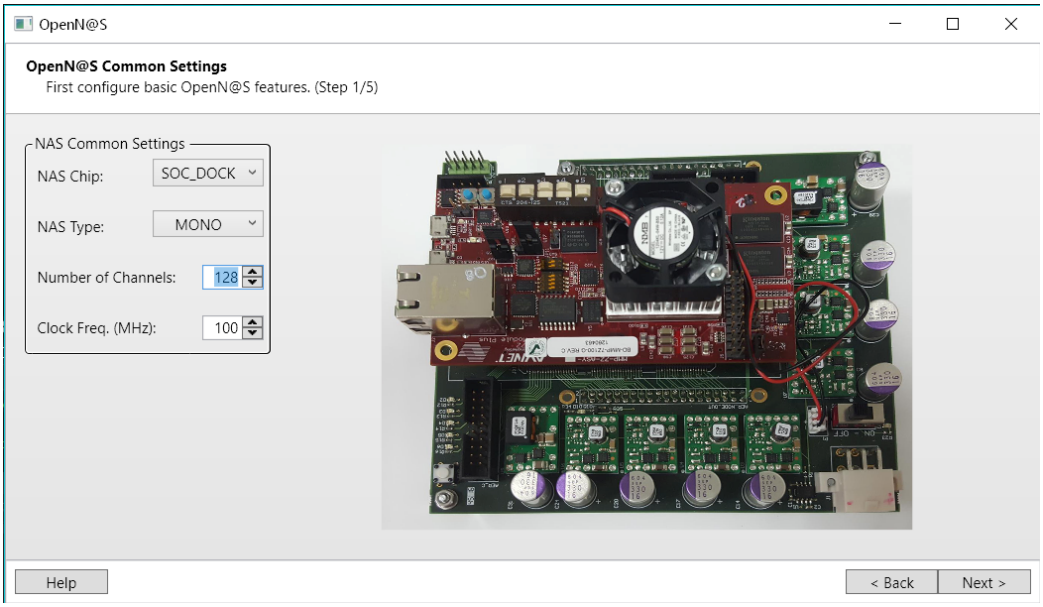

Figure 18: Setting the NAS chip to SOC_DOCK board.

After the target board has been selected, and the common settings has been set according to your needs, it is time to select which audio input we will use. Clicking on "Next" button, we will go to the second step of the NAS wizard.

AC'97 audio codec is selected by default. Although it is not used in the AER-Node board, it was the first audio input device used by other Xilinx FPGA-based board. This option let you only to modify the spikes generation module clock divisor.

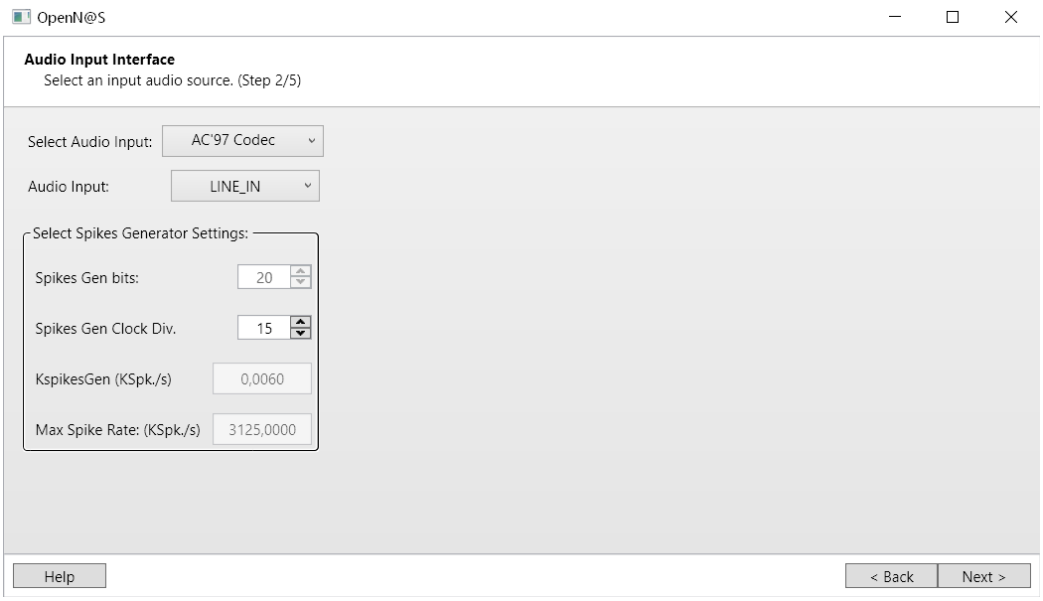

Figure 19: Audio input from AC'97 audio codec.

There exist a new version of audio input interface. It consists in a PCB which has a pair of PDM microphones and also a new ADC for audio signals. This PCB was designed to be used along with the AER-Node board. Then, if you decided to use AER-Node board, you will be able to use the audio codec. This option is called "I2S" because the codec use that communication protocol to transfer the data.

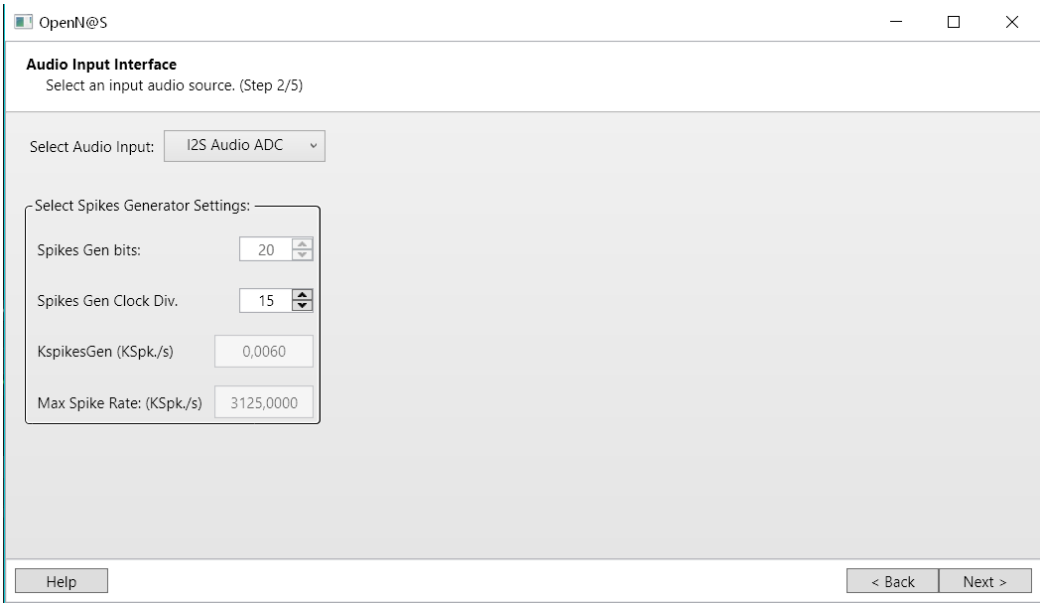

Figure 20: Audio input from line-in connector.

Also, as we mentioned before, you can select to use the PDM microphones. User can set a lot of parameters, since it was implemented a FPGA module to convert the microphones output to spikes using several spike-based high/low pass filters.

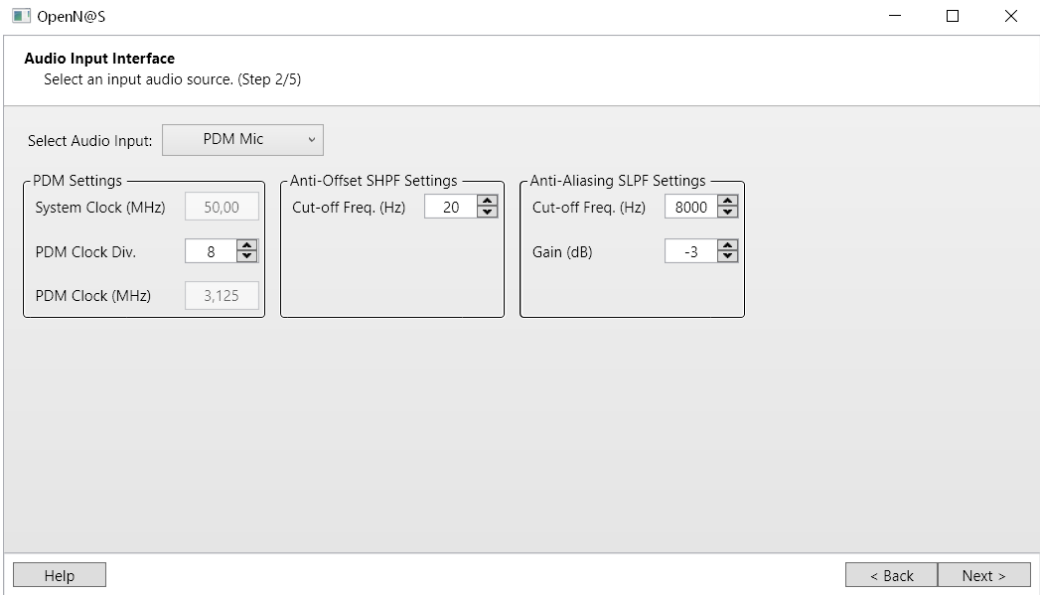

Figure 21: Audio input from PDM microphones.

Thus the audio input interface board implements two different sound input devices, and the process of programming file is quiet long, the tool offers you the possibility of select both inputs, and to select which one would you use by using a jumper placed on the PCB.

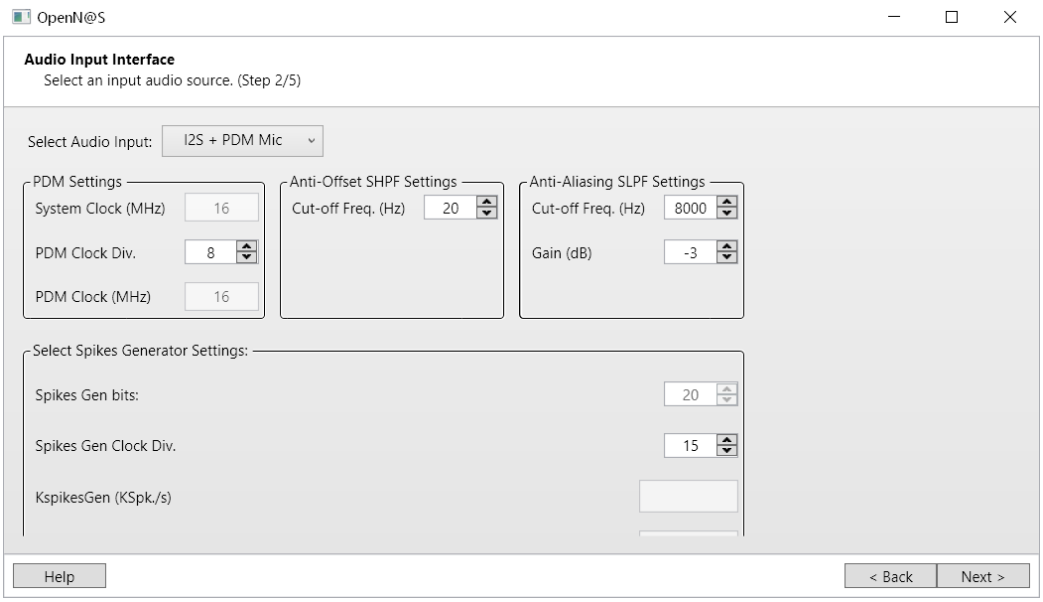

Figure 22: Audio input from either line-in connector or PDM microphones, selected by a jumper position.

Now, let's begin to configure the most important part of the NAS model. The original NAS's filters architecture is in cascade, according to the biology. So, by default this option is selected. Yo can establish a frequency range, which is defined by default with the human audible sound range (from 20 Hz to 22 KHz). A critical parameter is the SBPF Output Att. (dB), due to this parameter modify the filters output attenuation, that means it controls the amount of spikes that we will have in the NAS output. By decreasing the value, less spikes will be obtained at the output. Actually, we are working with values close to -30 dB.

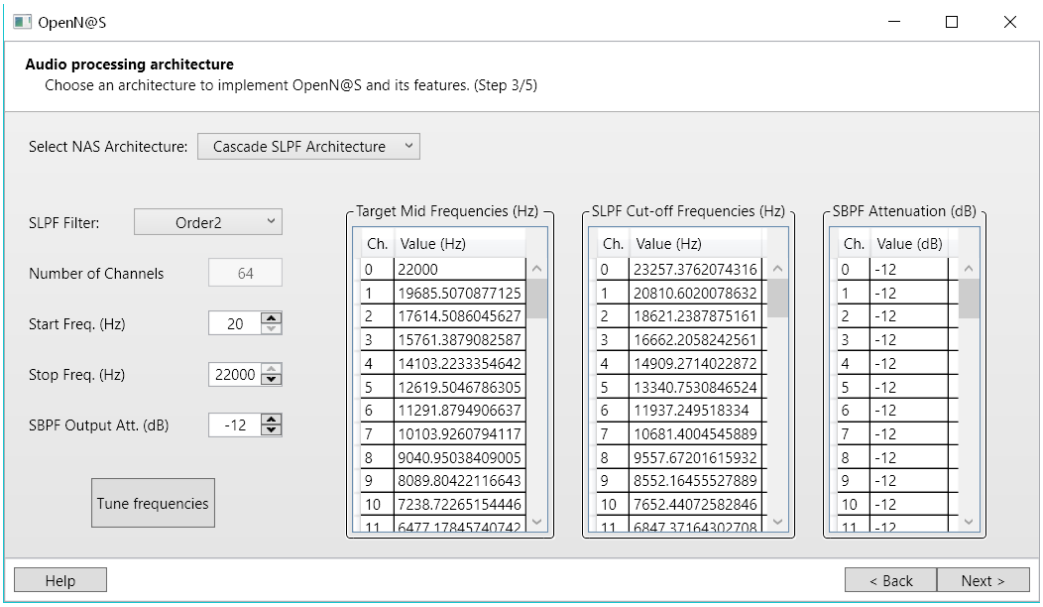

Figure 23: Selecting the NAS architecture. Cascade architecture is mostly used, and it is by default in the OpenNAS.

Filter's channels frequencies are calculated automatically by the software based on the specified architecture by the user. However, if you modify the attenuation value, you need to click on "Tune frequencies" button to recalculated all the parameters.

There is another NAS architecture approach, the parallel filter bank architecture. The parallel architecture is still under develop, but if you are curious you can take a look and try to use it.

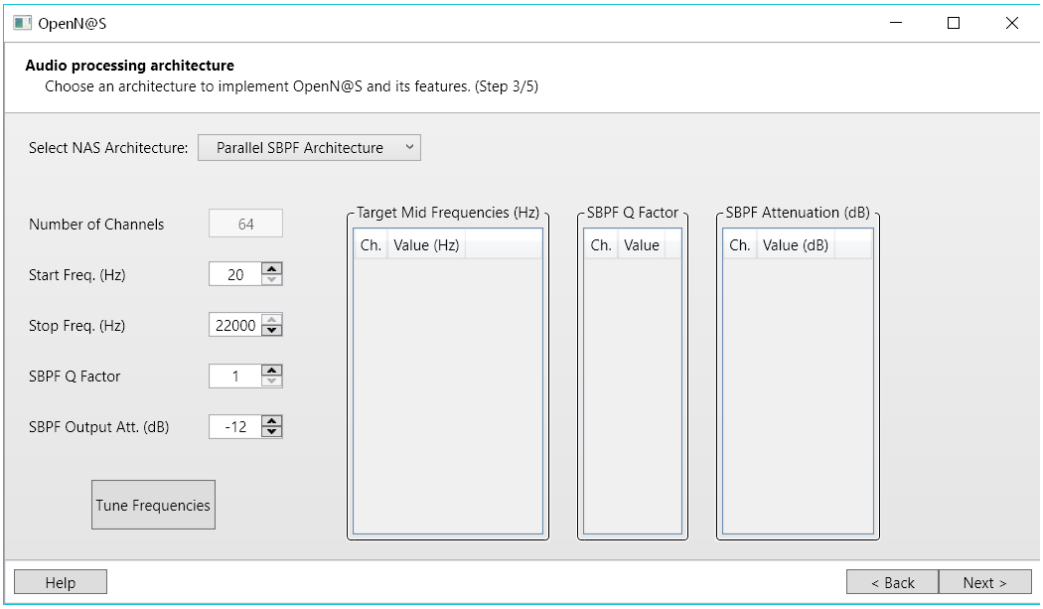

Figure 24: Parallel architecture open a different approach to the audio processing.

Almost the end, but not less important. It is time to choose the output interface. This step is completely dependent of what do you want to do by using the NAS. According with our experience, we recommend first to select the Spikes Distributed Monitor, which allows you to visualize the NAS output in real-time using the jAER software. Then, you can perform some recordings with your audio input dataset, and to analyze it by using the NAVIS tool, specifically designed and implemented for that purpose. NOTE: it is not recommendable to change the parameter values unless you need it.

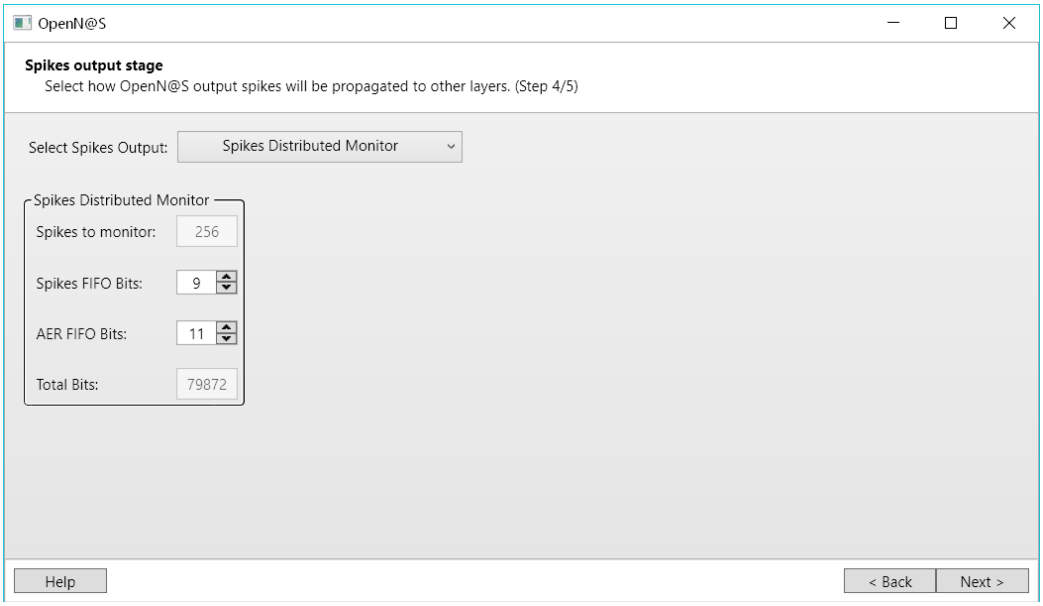

Figure 25: Using the AER Distributed Monitor as output interface allow you to monitor the NAS output in real time..

After checking the NAS output, and to be sure that the output will be useful for you, OpenNAS allows you the possibility of to connect the NAS with the SpiNNaker machine (both 4-chips and 48-chips machines). There are two versions of the AER-SpiNNaker interface; the v1, which is able to send spikes from the NAS to the SpiNNaker machine through the SpiNNaker link (using an adapter board). Any parameters are not needed in this case.

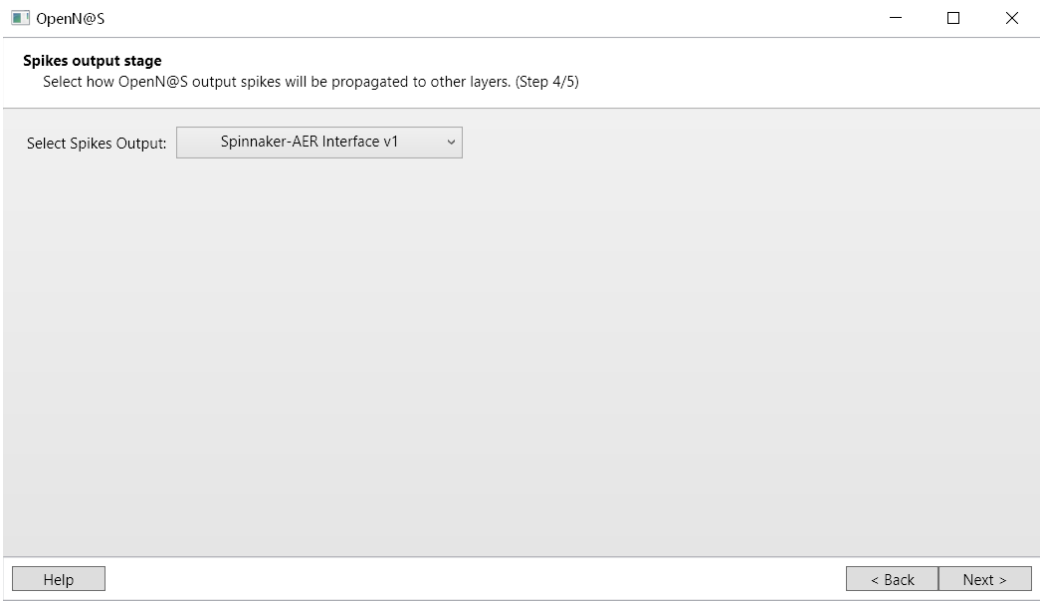

Figure 26: With SpiNNaker interface v1 you can send AER events from the NAS to the SpiNNaker machine.

The second version of AER-SpiNNaker interface is an adaptation of the work performed by Luis Plana to connect AER devices with the SpiNNaker machine in real time allowing bi-directional communication. Here you are the GitHub link of the original repository. There is also a complete documentation which we used to modify the original version in this link.

And, why don we not have a mixed output between AER monitor and SpiNNaker? Of course, we have implemented a version in which you can monitor the NAS output that are being sending to the SpiNNaker machine using jAER. But it could be a problem, because for each output spike, you will need the ACK signal from both jAER application and SpiNNaker machine. Hence, the system could work slowly.

The last step, after spend like... 2 minutes setting the OpenNAS is to generate the NAS files according with your parameters. You need to specify a folder in which all the files will be copied.

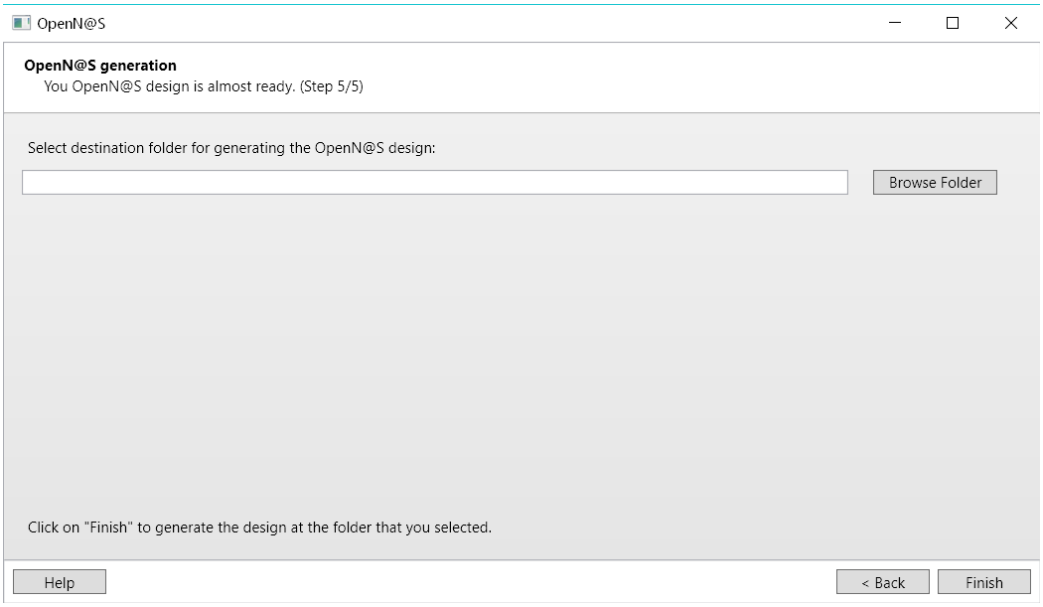

Figure 27: Select a destination folder to save the NAS files.

We recommend to create a very simple folders architecture: the main folder, named for example NAS_PDM_64ch_MONO_Monitor, and one folder to the source files, named "sources", and another one in which the ISE project will be created, named "project". Once the destination folder is selected, just click on the "Finish" button.

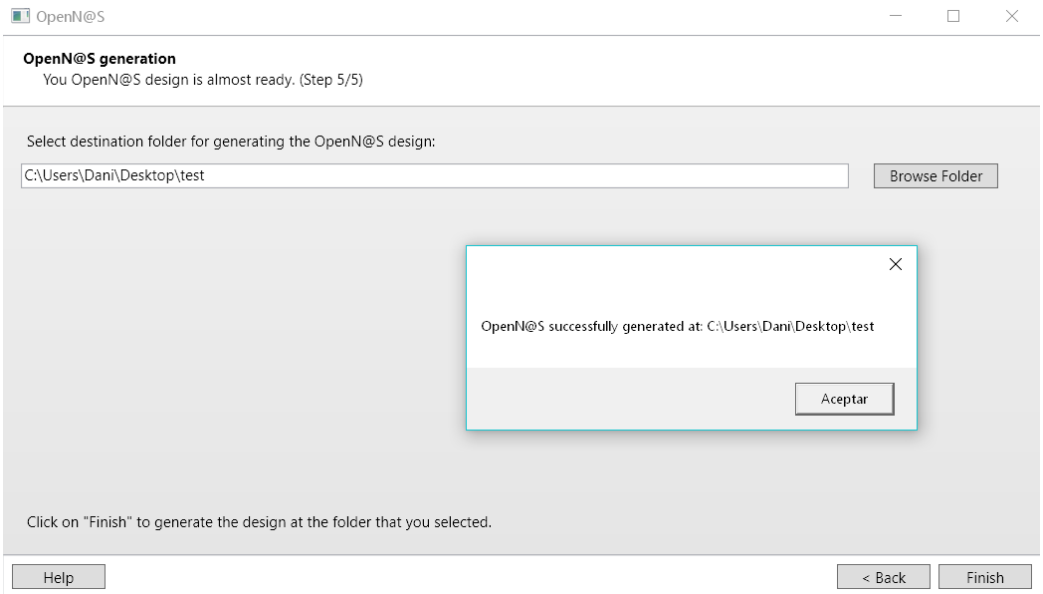

Figure 28: Files were generated successfully.

If everything is OK, a message window should appear telling you "OpenNAS successfully generated" and also showing you the destination folder. At this point, NAS HDL files has been generated taking into account your preferences, and you can take a look of the source code.

Anadir vista de los archivos que genera y que son cada uno. Hablar del xml.

3.3 Creating a Xilinx ISE project

By now, a lot of things have been taking into account to generate correctly your NAS model. And in this moment, since we already have the NAS VHDL files, you can decide what will you do with those files. You could want just load the files into the FPGA, or maybe you need to add new VHDL modules or modify the NAS VHDL modules to test something... In any case, you will need to create a new ISE project to, finally, generate the programming file with .bit extension and then to program the FPGA.

If you have an AER-Node board, SOC_DOCK board or ZTEX 2.13 board (OpalKelly will be available soon), and you do not want to add/change any VHDL modules, you can directly get the .bit file by running a .tcl script by

using the ISE Design Suite 32 Bit Command Prompt. (Still under development, but it will be available soon!).

Meanwhile, of if you do not have one of those boards, you will need to create a ISE project manually, to configure it correctly and to add the source files by hand. First step, of course, is open the ISE 32-bit Project Navigator. The icon looks like is showed in the next Figure 29:

Figure 29: ISE Project Navigator icon.

Left click on the icon twice, and then the project navigator will be opened. Next image show us how the program looks like when it is started.

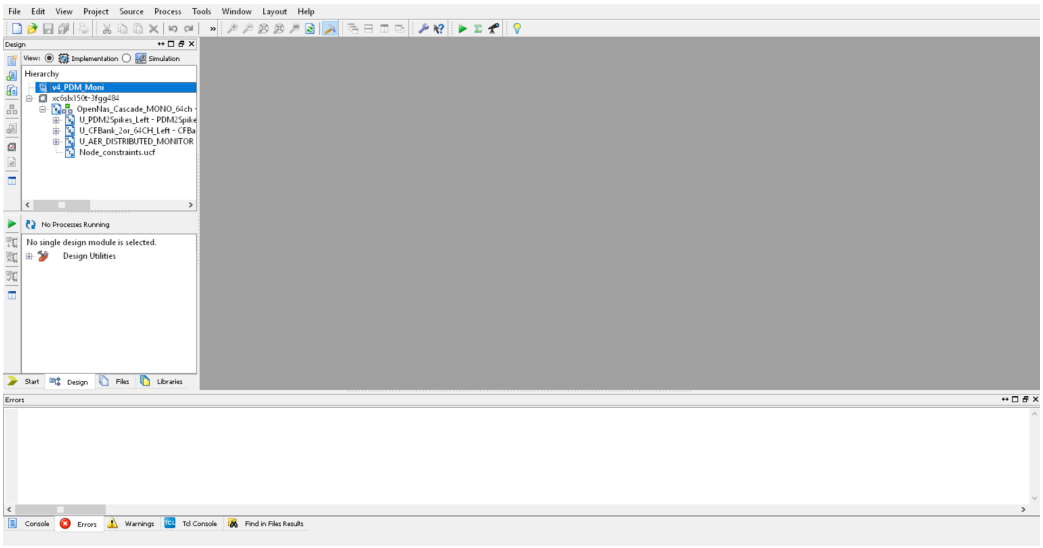

Figure 30: ISE Project Navigator main view.

So, we need first to create a new project. To do that, go to File, and

select New Project.

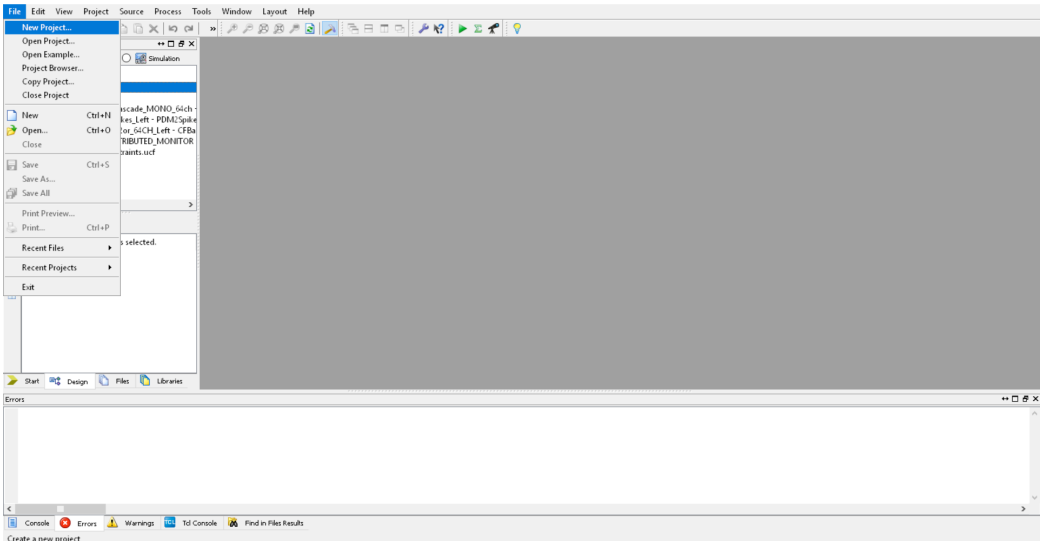

Figure 31: Creating a new project in ISE Project Navigator.

Then a new window appear. This is the New Project Wizard, and it is easy to follow. Let's start it.

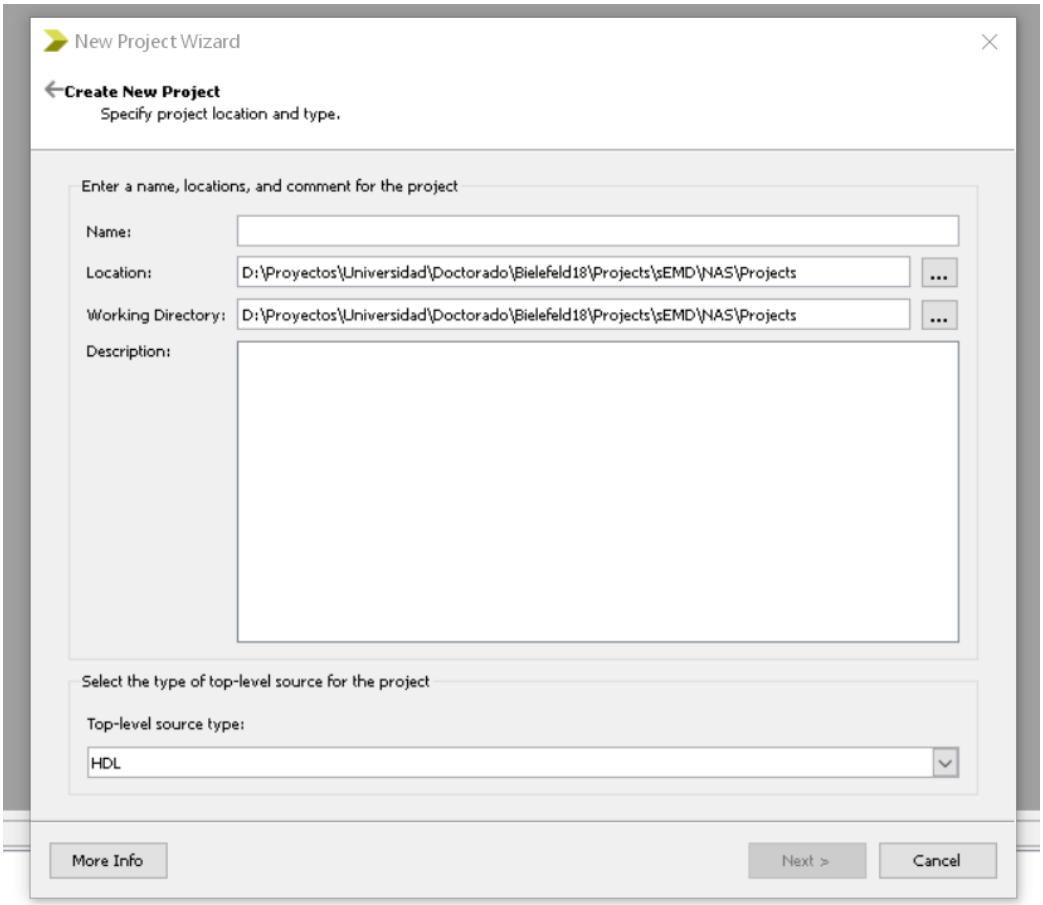

Figure 32: ISE new project wizard.

Like in every new project, you have to indicate the project name. According to the folders architecture aforementioned, we recommend to select the same name used by the sources folder. In our case, our project will be named NAS_PDM_64ch_MONO_Monitor. And the location will be ../NAS_PDM_64ch_MONO_Monitor/project/. Now, click on the "Next" button.

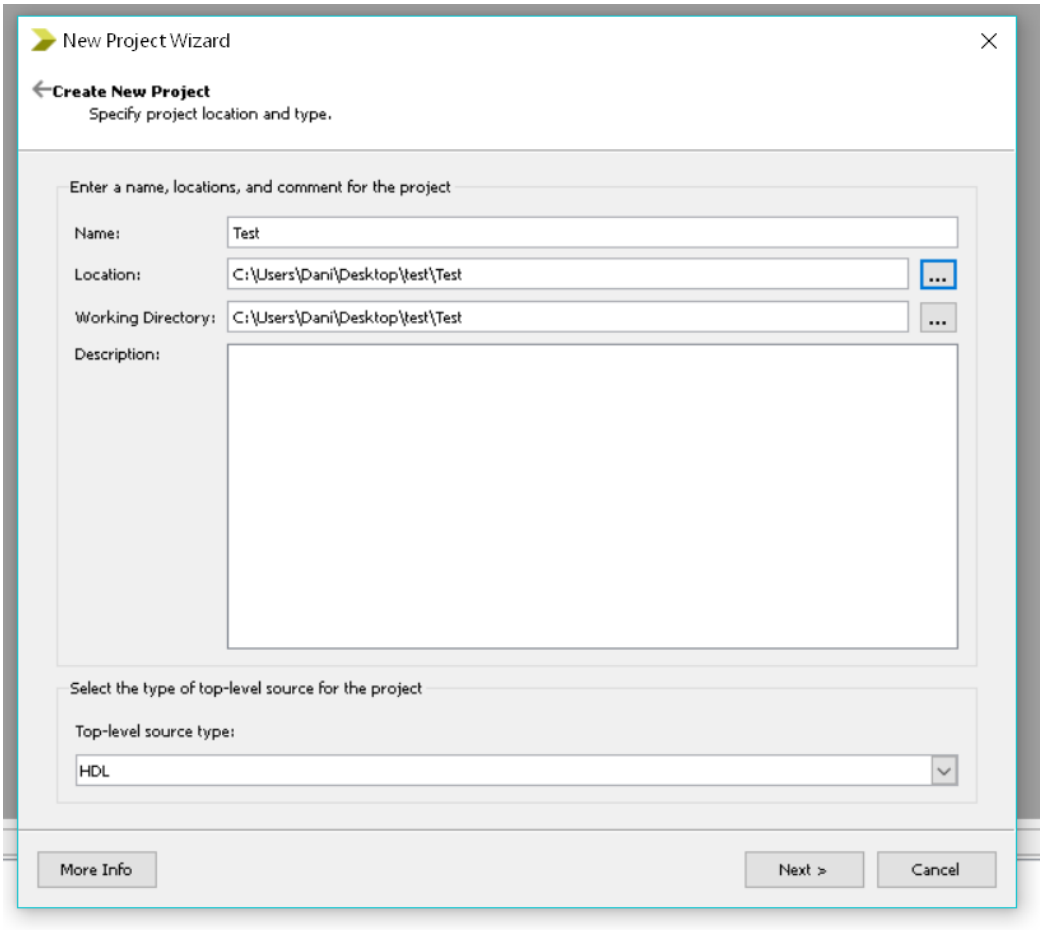

Figure 33: Setting the ISE project name and the project localization.

CAUTION! This step should be done carefully. In the project settings step, the FPGA chip must be selected. Make sure that all the fields have the same values that the showed in the image. Those values are only valid if the AER-Node board is used. For others board, user should make its own project settings.

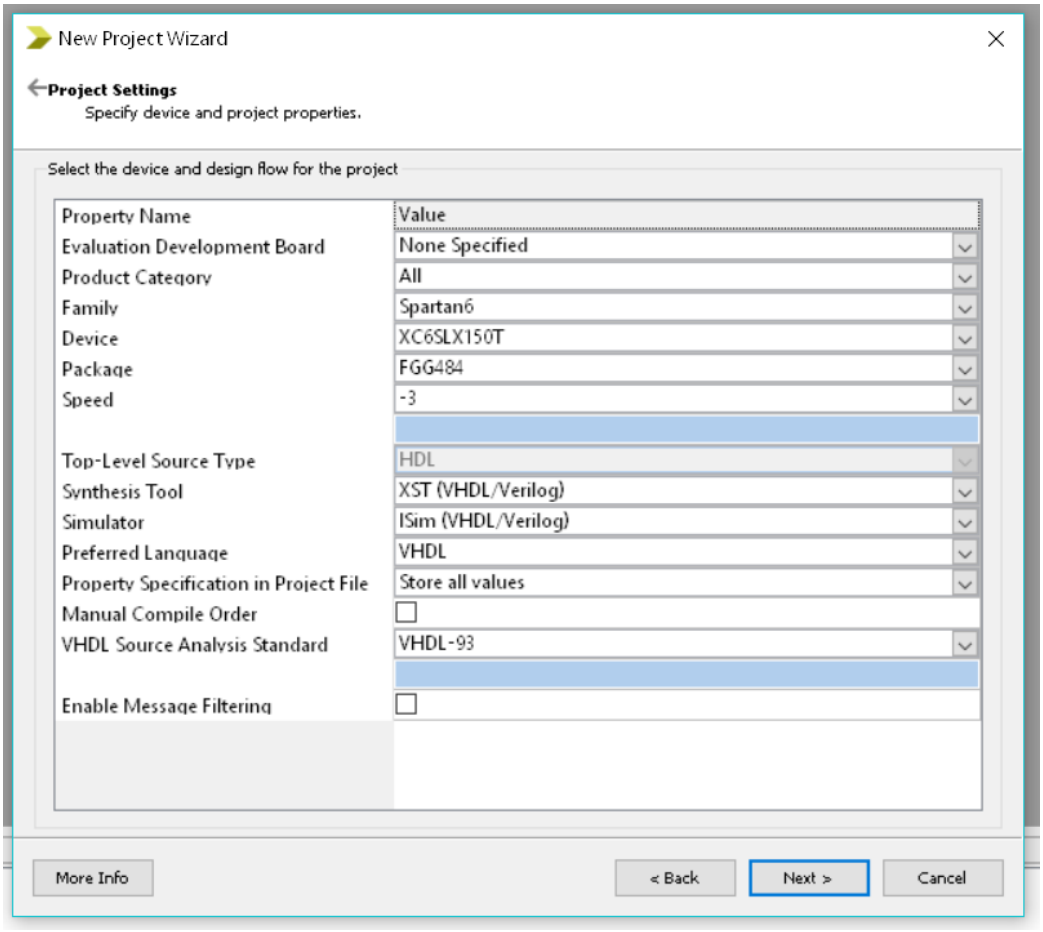

Figure 34: Correct project settings to create a ISE project for the AER-Node board.

At the end of the wizard, a summary of the new project is showed. Check this summary to prevent future errors. If everything is correct, click on "Finish".

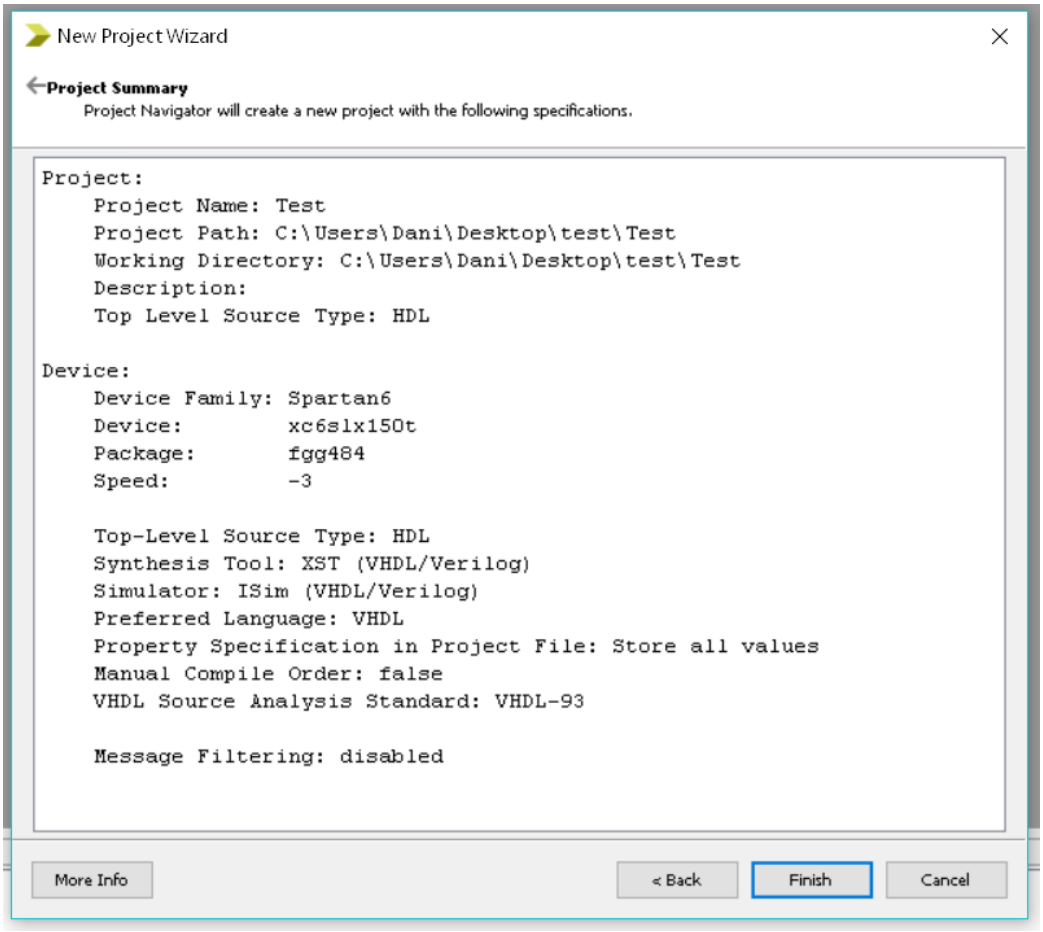

Figure 35: ISE project settings summary.

Our project is now created, and we can see in the top left a new folder with the name of our project, and the selected FPGA chip.

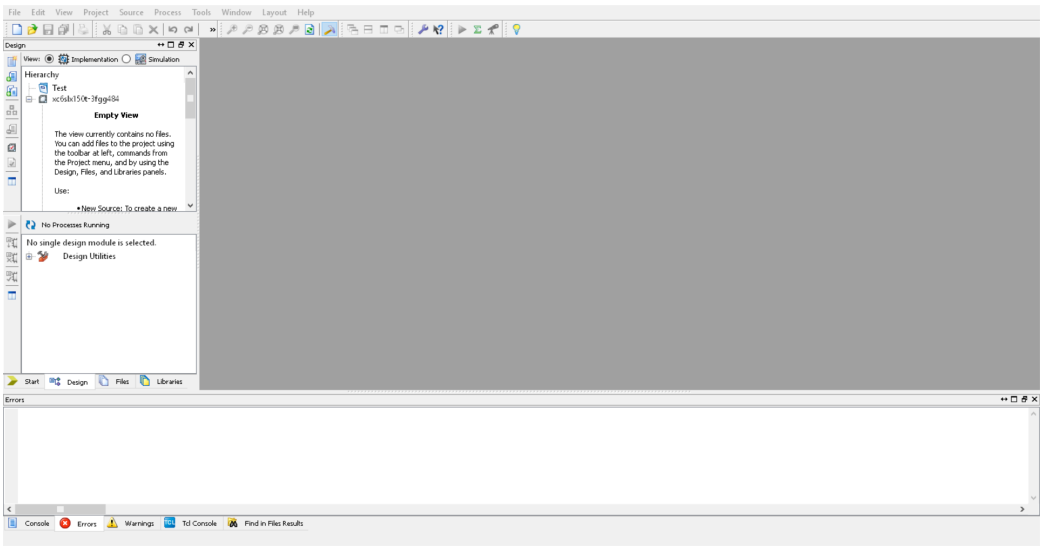

Figure 36: Main view after create a new project.

And now, the last "very important" step, which is to add the NAS VHDL files generated by the OpenNAS into the new ISE project. But, why is this an important step? There are many reasons, but the two more important reasons are: because you need to add the correct files, and because is recommendable to add those files, not to add a copy of those files. Let see. First, make right click either on the project folder or on the selected chip. A menu appears, and then select "Add sources".

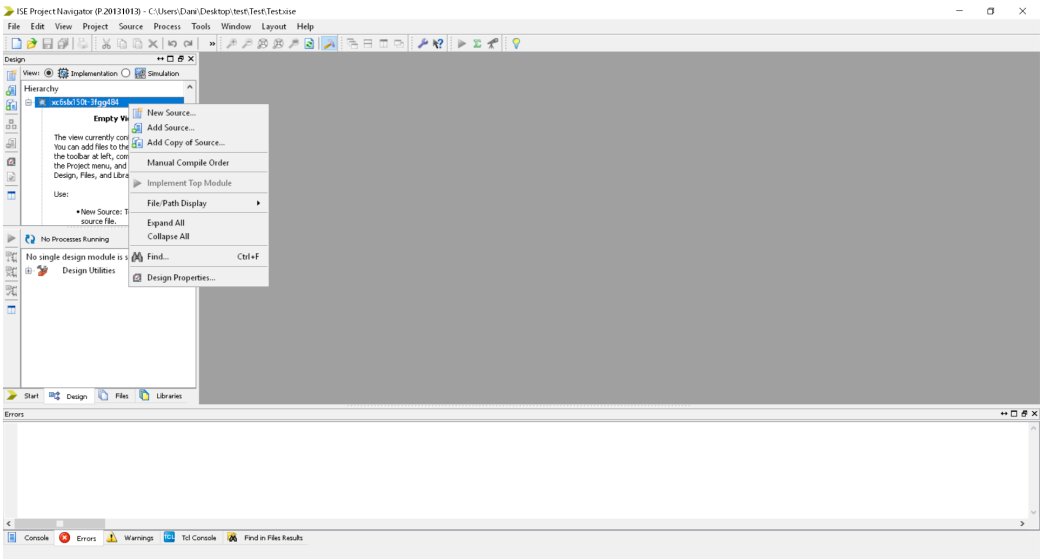

Figure 37: Ready for adding the NAS VHDL sources.

An emergent windows should appear to navigate where the NAS VHDL sources are. But be careful, if it is not the first project that you have created, then you may select the wrong OpenNAS sources with different parameters. If you follow our recommendation of how to organize the NAS projects, the source files should be in ../NAS_PDM_64ch_MONO_Monitor/sources/.

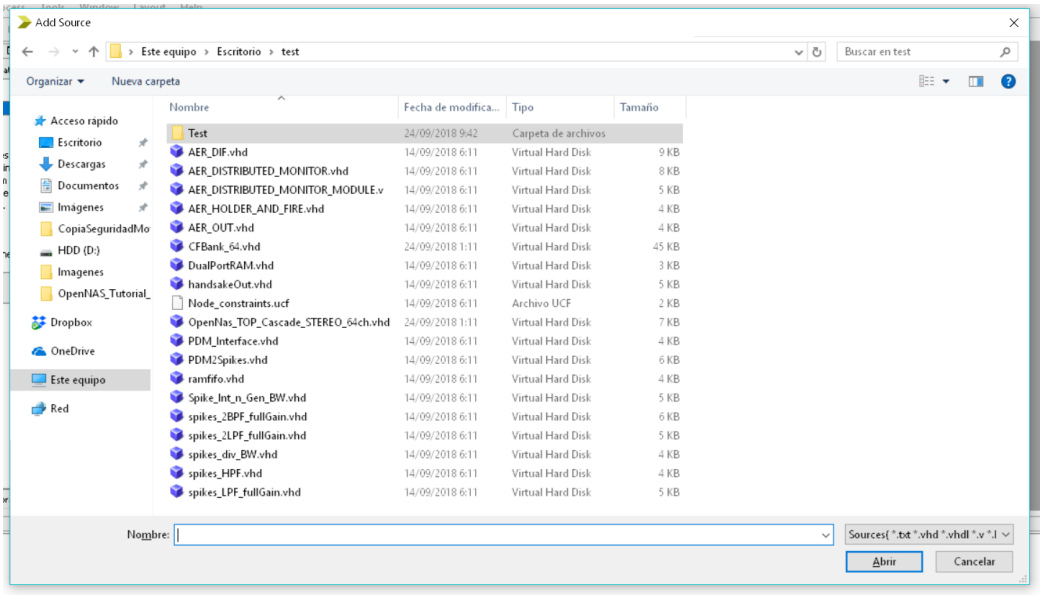

Figure 38: Go to the destination folder in which the NAS VHDL were saved.

When there, just select all the files, and click on "Open" button to add them.

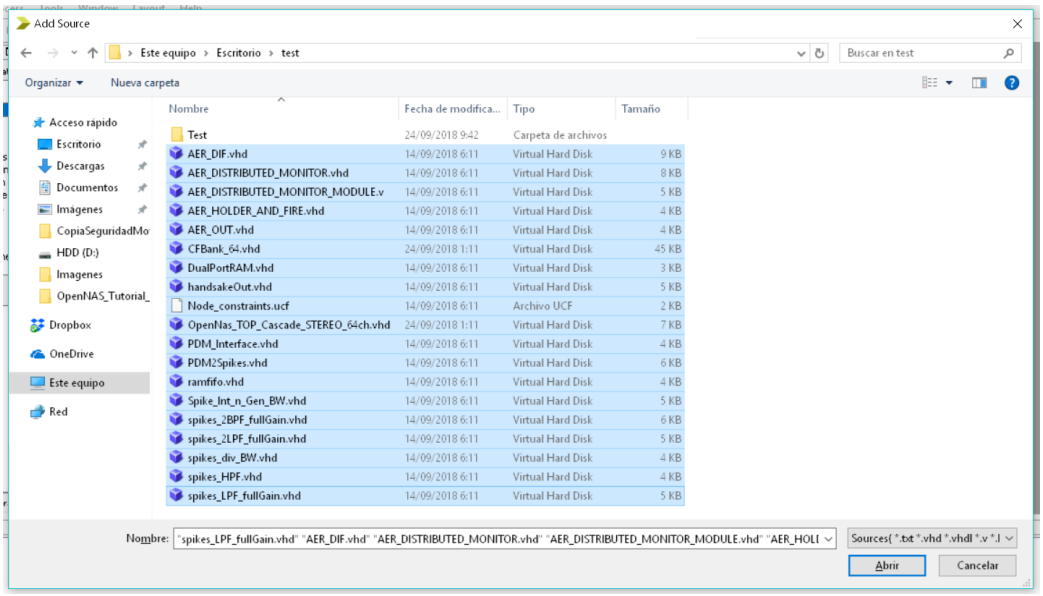

Figure 39: Select all the files, except the .xml file.

In a ISE project, source files can be included for simulation, for implementation, or for all. If we take a look, by default all files have the field "Association" to "All" except the constraint file .ucf, which does not make sense in a simulation process. We do not need change any values. But if you want to add a testbench file, the file association should be "Simulation". Click on "OK" to finish this step.

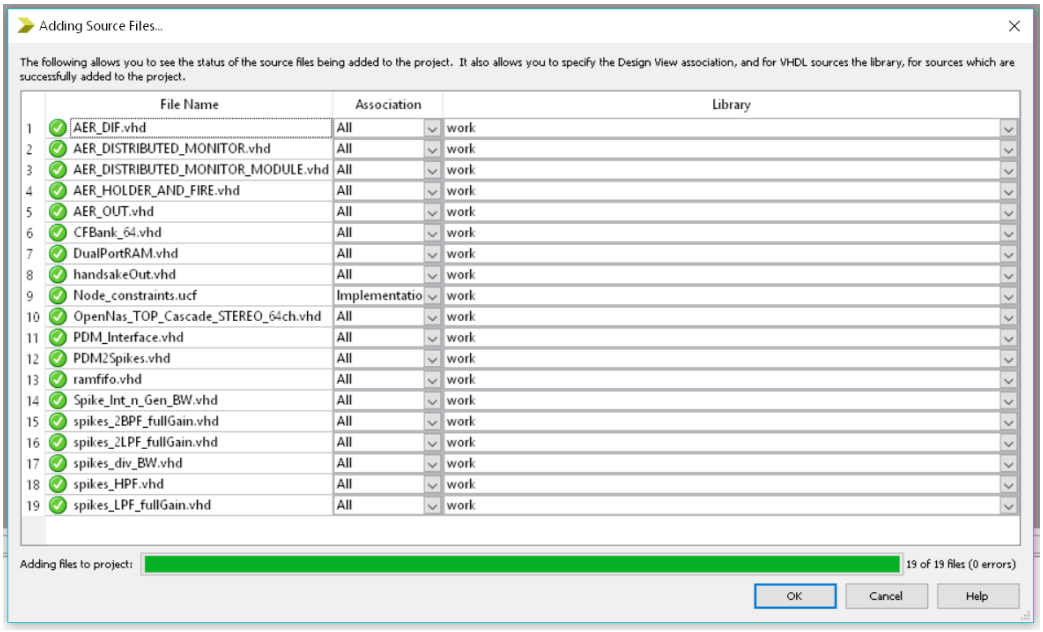

Figure 40: Selecting the files association.

Checking again the project folder in the ISE main window, now we should have a new hierarchy with the included files.

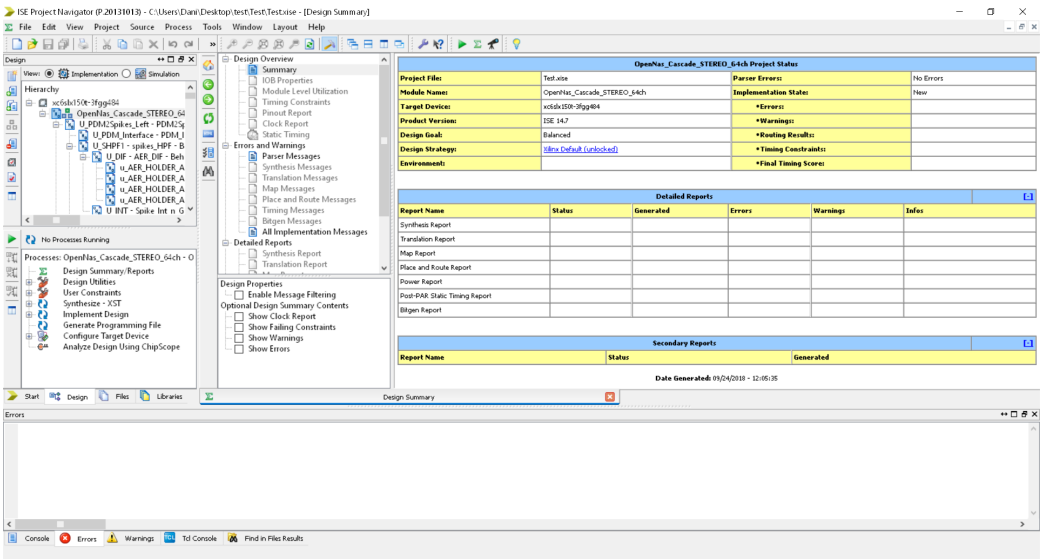

Figure 41: ISE main window after to add the source files.

By analyzing that hierarchy, it is easy to identify the three different parts in which we split the NAS architecture: the input (U_PDM...), the processing block (U_CFBank_2or_64CH), and the output (U_AER_Distributed_Monitor).

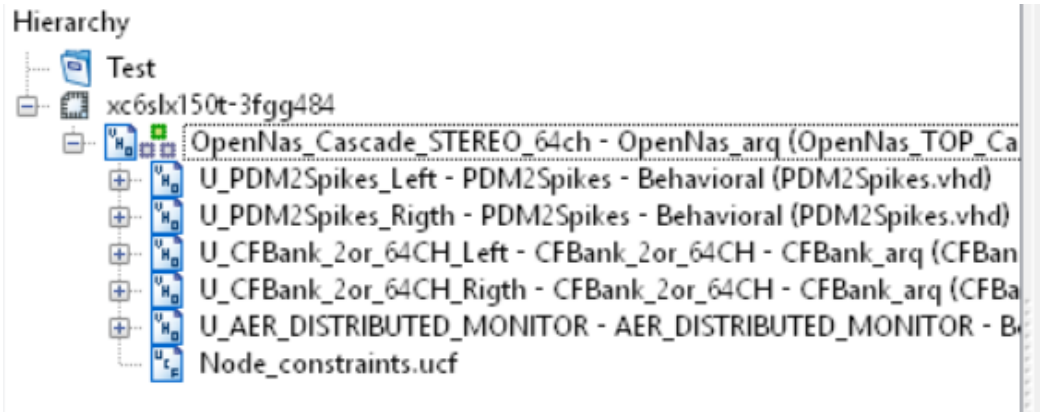

Figure 42: OpenNAS project modules hierarchy.

Finally, to start the process of programming file generation, we have to make left click on the top file of the VHDL files hierarchy, and just below some options will appear.

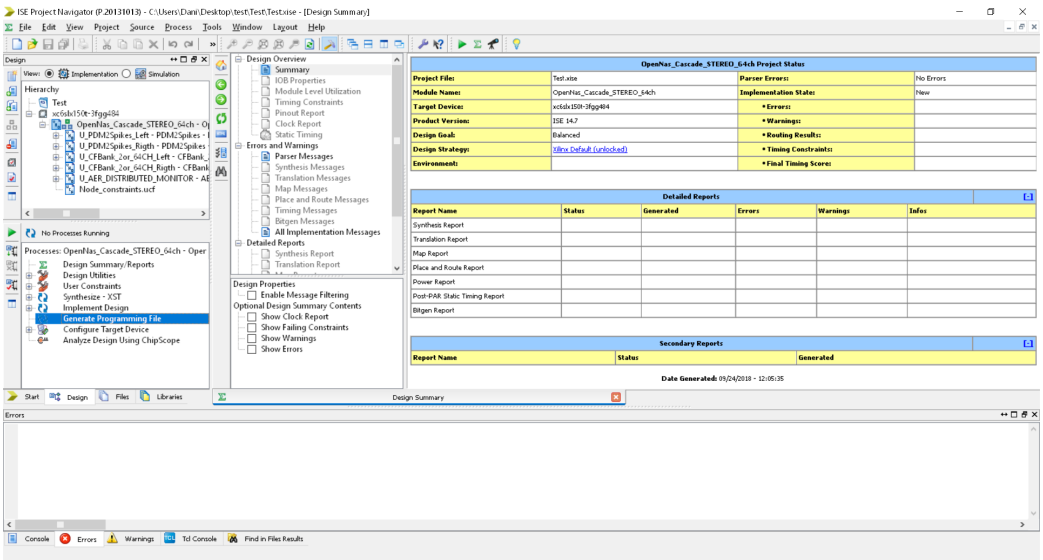

Figure 43: Clicking on the top file to show the process menu.

Just making double left click on the Generate Programming File option, the entire process will start to running, starting from the synthesis process, implement design process and, at the end, the programming file generation. When the process is finished, you can find the programming file in the project directory, with the extension .bit.

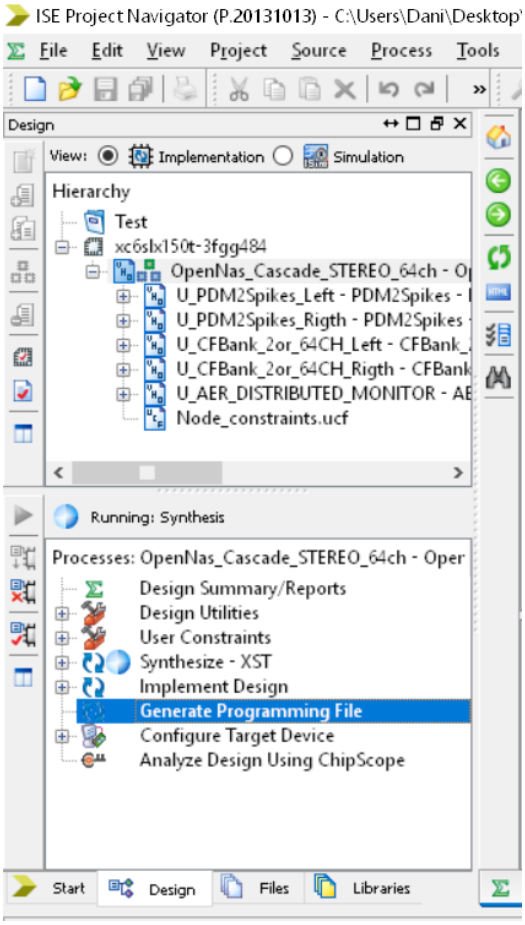

Figure 44: Running the Generation Programming File process.

Say "need to wait".

3.4 Loading bitfile into the AER-Node board

Once the .bit file has been generated, you need to load it into FPGA. Click on this NAS setup getting started link to carry out that process. Moreover, to perform the post-processing step, visit NAVIS GitHub website [1].

4 Some tricks!

- If you select SpiNNaker v1 as output interface, it would be recommendable to reset both boards in a specific order: push reset button from both boards, then leave the reset button from SpiNNaker, wait until the Ethernet LED's start to blink, and finally leave the reset button from AER-Node board.

References

[1] J. P. Dominguez-Morales, A. Jimenez-Fernandez, M. Dominguez-Morales, and G. Jimenez-Moreno. Navis: Neuromorphic auditory visualizer tool. *Neurocomputing*, 237:418–422, 2017.

[2] A. Jiménez-Fernández, E. Cerezuela-Escudero, L. Miró-Amarante, M. J. Domínguez-Morales, F. Gomez-Rodriguez, A. Linares-Barranco, and G. Jiménez-Moreno. A binaural neuromorphic auditory sensor for fpga: A spike signal processing approach. *IEEE Trans. Neural Netw. Learning Syst.*, 28(4):804–818, 2017.

[3] A. Jimenez-Fernandez, A. Linares-Barranco, R. Paz-Vicente, G. Jiménez, and A. Civit. Building blocks for spikes signals processing. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[4] R. F. Lyon and C. Mead. An analog electronic cochlea. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1119–1134, 1988.