University of Sevilla

Robotics and Technology of Computers Lab.

Architecture and Technology of Computers Dept.

OpenNAS tutorial

Authors:
Daniel Gutierrez-Galan
Juan Pedro Dominguez-Morales
Angel Jimenez-Fernandez
Alejandro Linares-Barranco

June 13, 2019

Abstract

OpenNAS is an open source VHDL-based Neuromorphic Auditory Sensor (NAS) code generator capable of automatically generate the necessary files to create a NAS-VHDL project for FPGA. OpenNAS guides designers with a friendly interface that allows configuring the NAS using a five-steps wizard for later code generation. It includes several audio input interfaces (AC'97 audio codec, I2S ADC and PDM microphones), different processing architectures (cascade and parallel), and a set of neuromorphic output interfaces (parallel AER, Spinnaker). After NAS generation, designers have everything prepared for building and synthesising the VHDL project for a target FPGA.

Contents

List of Figures

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), a maximum output event rate of 2.19Mevents/sec, a power consumption of 29.7mW, slices requirements of 11,141 and system clock frequency of 27MHz."

Latter paragraph is the abstract to the NAS [2] original paper. At the beginning, the idea was to design, develop and test spike-based 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 classical methods. One example of this simplification could be the spike integrate & generate, which mimics the integrate & fire neuron behavior. 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].

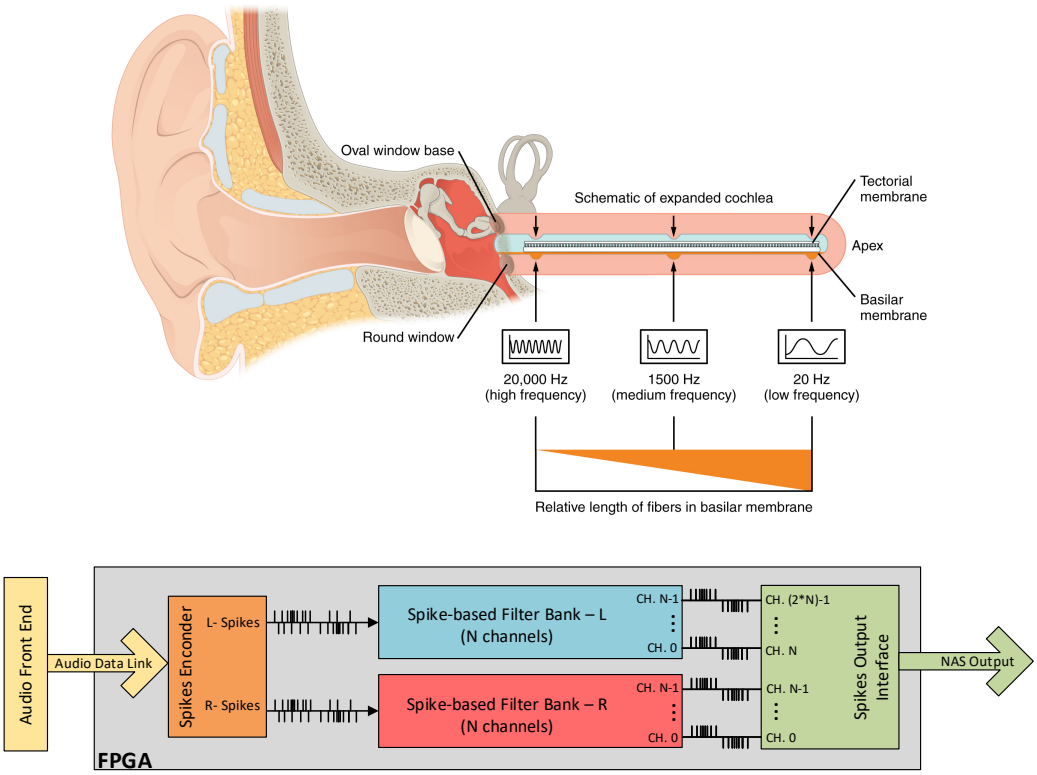

Figure 1: Top: biological model of the human ear. Bottom: NAS architecture block diagram.

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

2 OpenNAS overview

2.1 Wizard overview

Once OpenNAS tool has been executed, and the Welcome screen appears, the user only needs to complete the five steps of the OpenNAS wizard to obtain the generated VHDL files.

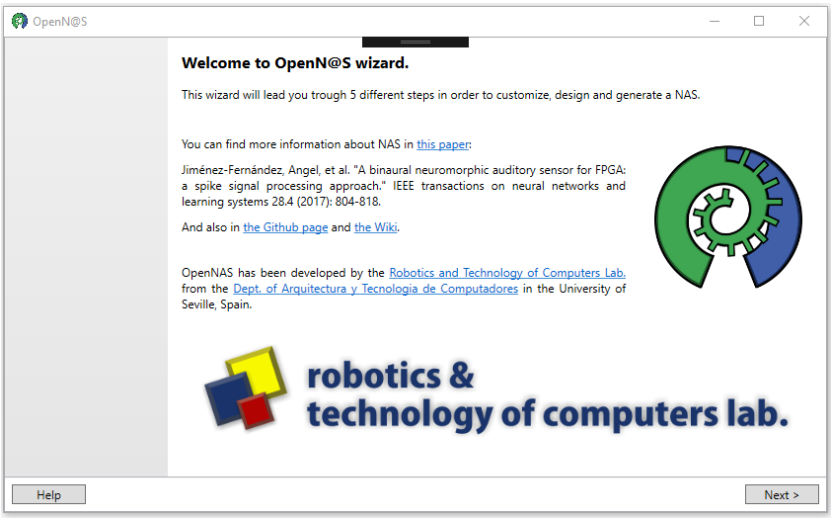

Figure 2: Welcome screen.

The Welcome screen shows a brief text which indicates to the user what OpenNAS tool does and also the information about our research group. Click on Next button to move forward in the wizard.

Figure 3: Selecting commons settings.

The first step allows to the user to select the NAS common settings, which are the target FPGA chip and its clock frequency, if the NAS is MONO or STEREO, and the number of frequency channels. A picture of the FPGA-based board selected in NAS chip is shown to the user to quickly identify the needed hardware components.

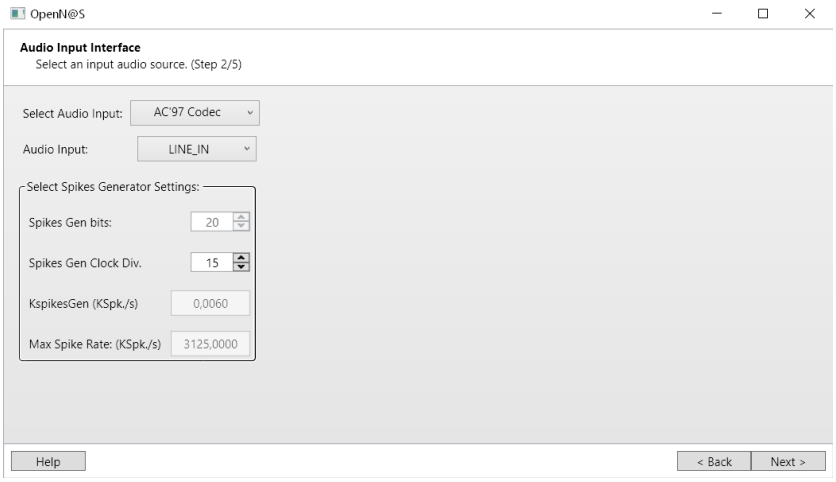

Figure 4: Selecting the input interface.

In this step, the input audio source must be selected. There are several options, among which we can find the AC'97 audio codec, a pair of Pulse

Density Modulation (PDM) microphones, and an I2S-based audio codec. Each input option has its own configuration parameters, which the user can set according to its project requirements.

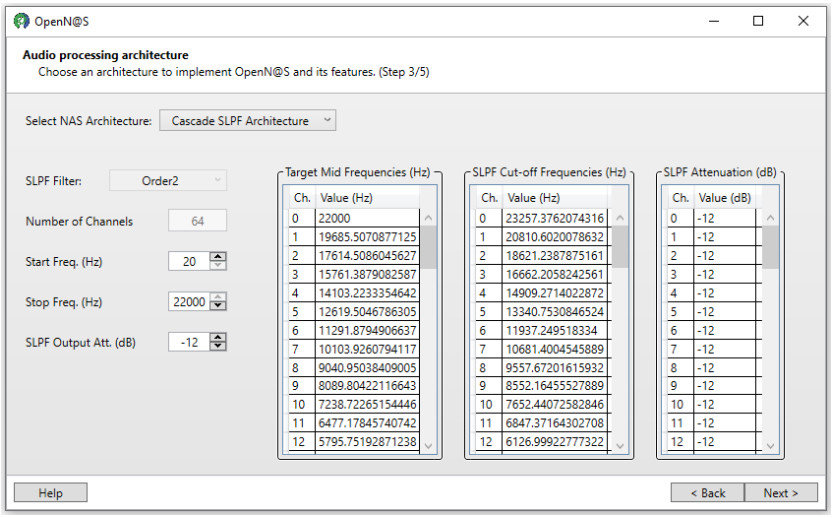

Figure 5: Selecting the processing architecture.

NAS processing architecture is defined in step 3, where the user can choose either a cascade or parallel architecture. Besides, filters order and filters output attenuation can be set. Finally, the user can define a frequency range between which the NAS will work, set by default as the human audible sounds range (from 20Hz to 22KHz).

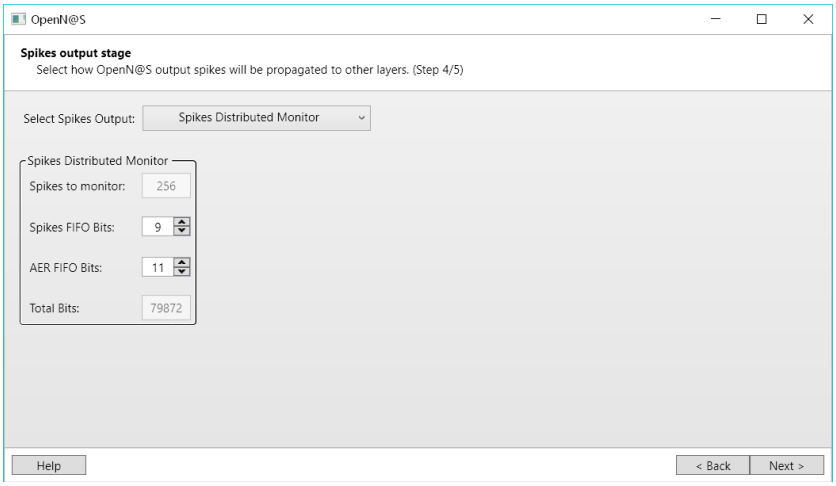

Figure 6: Selecting the output stage.

The output interface is selected in the fourth step. As in the most of the event-based neuromorphic devices, the AER protocol is used as output. For this reason, the AER monitor is used as output interface by default and then to connect our neuromorphic sensor to the application jAER and to be able to visualize the NAS output in real-time. However, it is also interesting to connect the NAS output to others neuromorphic hardware, as the SpiNNaker board, in which Spiking Neural Networks (SNN) can be deployed and it can get as input data the output spikes from the NAS.

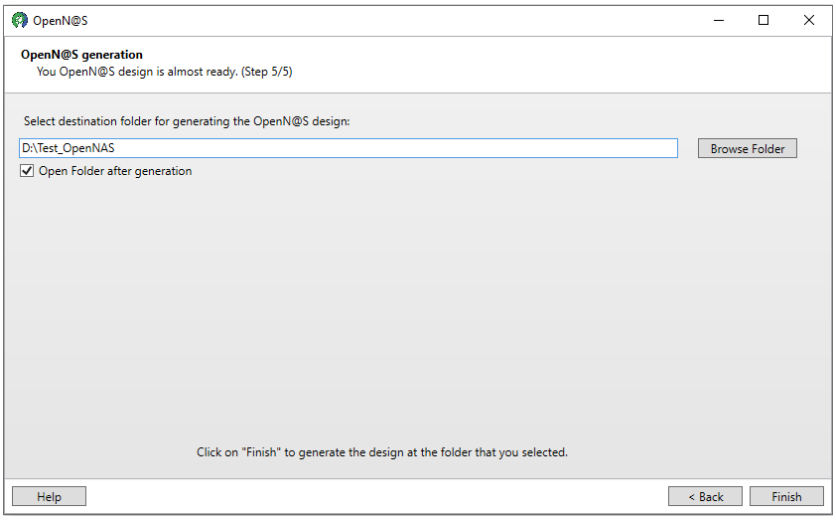

Figure 7: Selecting the destination folder.

Finally, when all previous steps have been done, the destination folder in which the VHDL files are going to be generated has to be selected by the user. It needs to create a new folder by hand, and then select it as the destination folder.

2.2 Requirements

2.2.1 Software dependencies

First, we need to check some software requirements to be sure that OpenNAS tool is going to work properly. All the software packages belong to the Microsoft .NET Framework. Therefore, we will need a **Microsoft Windows OS** version running on 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 a specific framework, which includes features used in our software tool. In this case, **.NET Framework 4.5** was used. 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 on the FPGA chip 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. 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

2.2.2 Hardware setup

About the hardware requirements, you can use your own boards, but we can not offer any guarantees about the system operation. You will need a FPGA-based board, at least 20 I/O pins, and either an audio codec on-board or interface with other audio input samples, as the I2S protocol. If not, an interface module must be implemented by the user. If you are interested to use our neuromorphic hardware, please visit the COBER website and check the neuromorphic hardware catalogue.

2.2.3 Requirements summary

- Software:
    1. Microsoft Windows OS: 7 or latest.
    2. .NET Framework 4.5.
    3. Xilinx ISE or Vivado.
    4. Xilinx iMPACT.
- Hardware:
    1. FPGA-based board.
    2. 20 I/O pins available.
    3. audio codec with I2S protocol or PDM microphones.

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: .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. Support are provided by our research group: **PLEASE**, let us know if you have problems with both the software and hardware dependencies/requirements. We will be happy to help you.

2.3 Usage flow

This section summarise the steps to follow in order to generate a NAS model as well as the files generated along the whole process, showed in Fig.8. During the process, two parts can be distinguished, First, the OpenNAS tool has to be used to generate the VHDL files. Once that, the user can use its preferred IDE for the next steps.

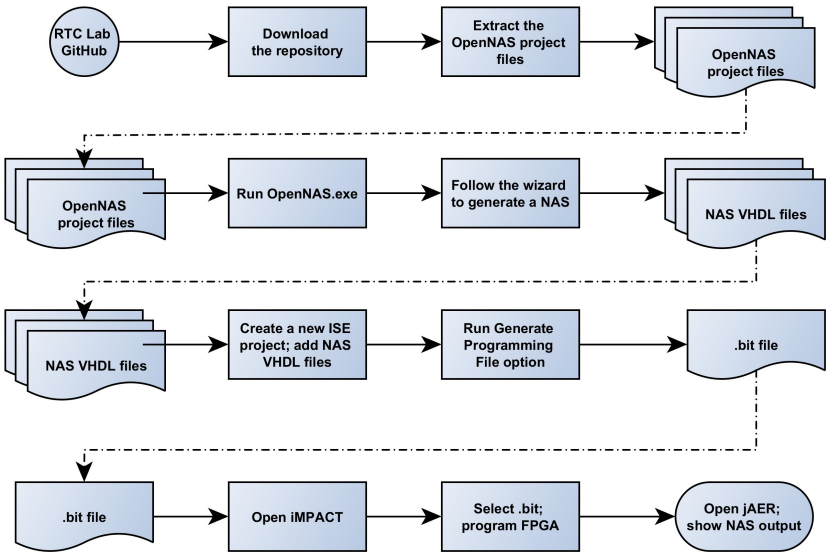

Figure 8: Complete usage flow to work with NAS, starting from the GitHub repository to showing the NAS output by jAER.

Each row of the Fig.8 is independent of the others. That means the user doesn't need to repeat every step every time you want to generate a NAS model. For example, you don't need to download the OpenNAS tool every time you want to use it. However, it is recommendable to check the repository for the updates.

2.4 OpenNAS options and parameters

Here we present a list with the available options and parameters that the user can set in the OpenNAS application. The list is sorted by following the wizard steps, i.e., following the order of appearance through the wizard.

2.4.1 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)

2.4.2 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.

2.4.3 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): from 0 to 50 KHz.
- Stop Freq. (Hz): from 0 to 50 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 Attenuation (dB)

2.4.4 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

2.4.5 Generated files

- Destination folder.
- Open folder.

3 OpenNAS wizard: how to use step by step

At this point, we are ready to start using the OpenNAS software tool. 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.

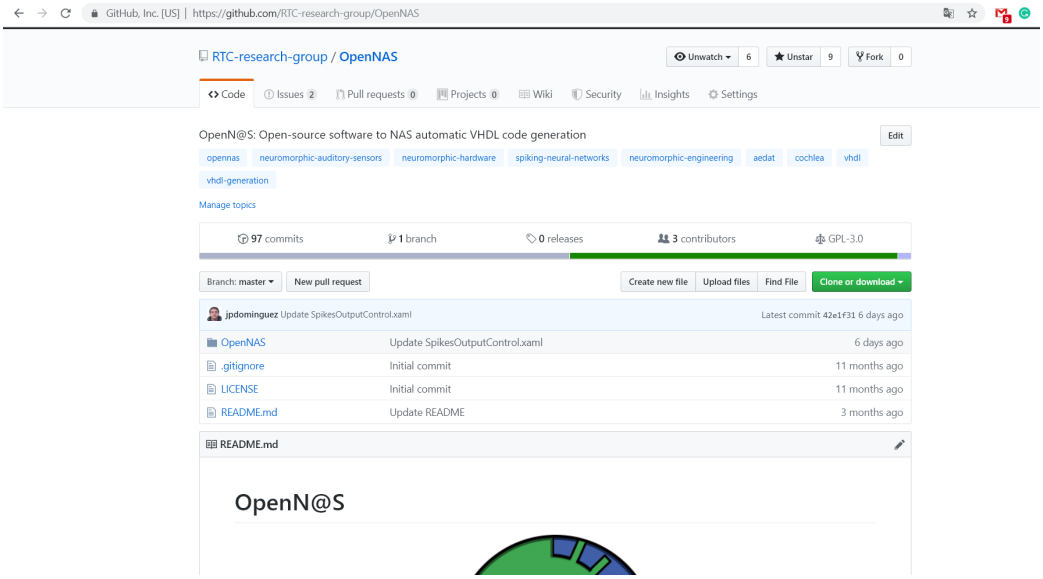

Figure 9: OpenNAS GitHub repository.

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 toolset to work with AER-based devices.

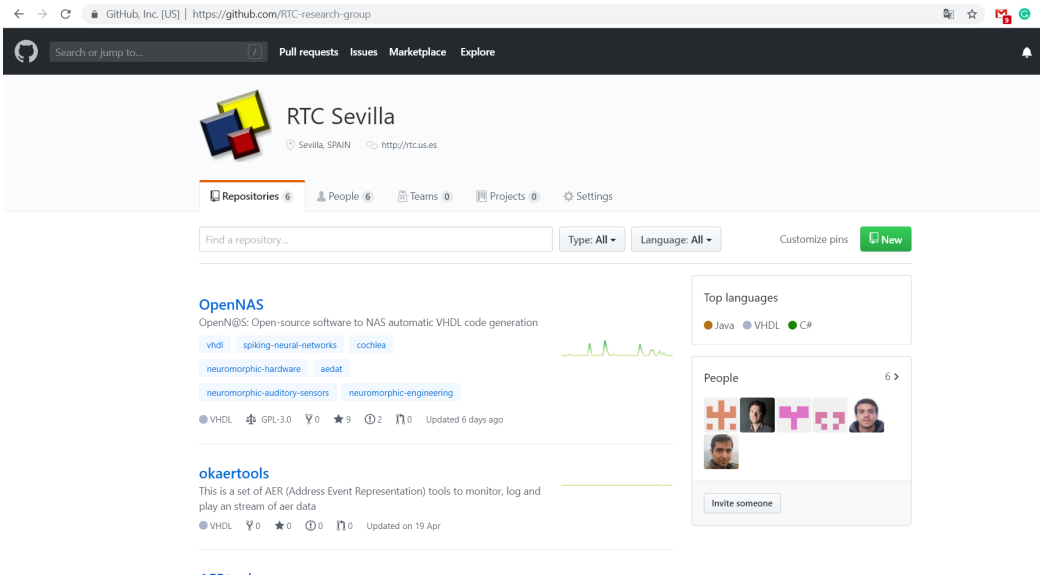

Figure 10: Robotics and Technology of Computers Lab. (RTC) GitHub page.

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

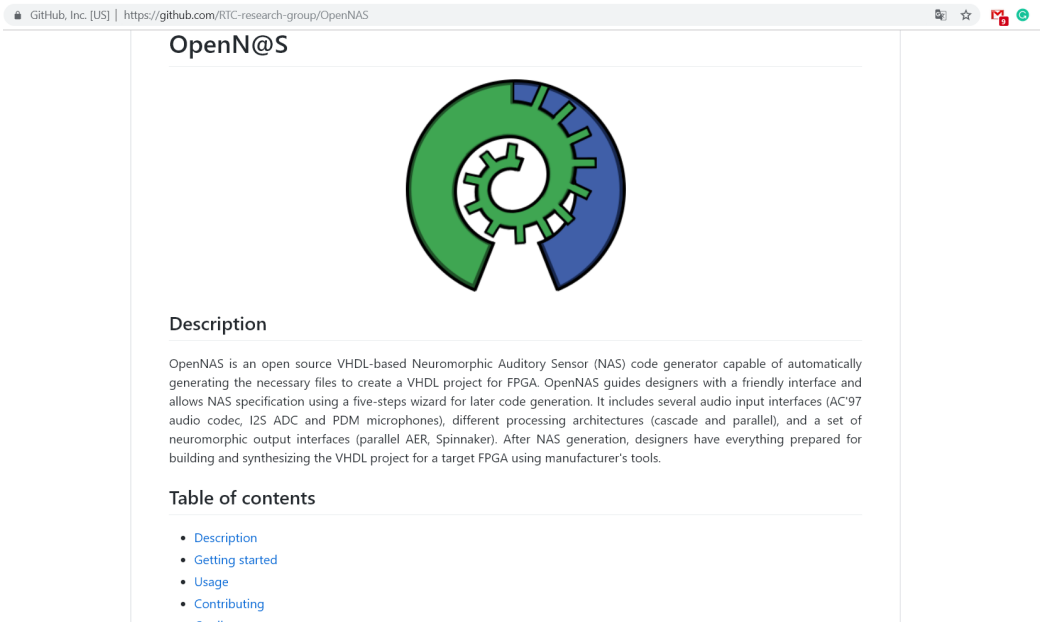

Figure 11: OpenNAS wiki overview.

The next step is either to download the project on 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!

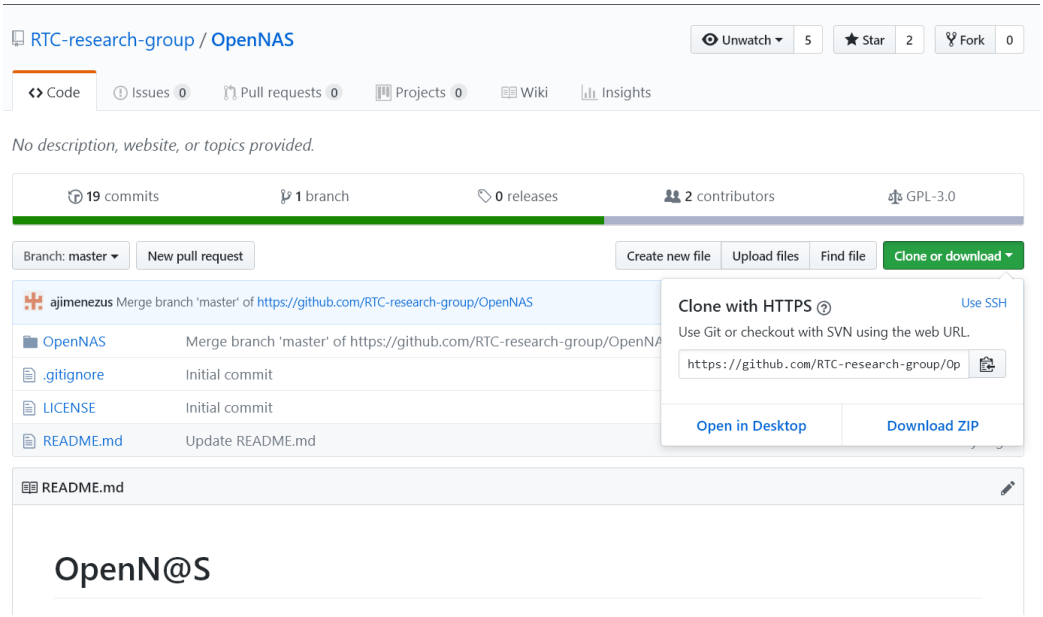

Figure 12: How to either clone or download the OpenNAS GitHub repository.

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".

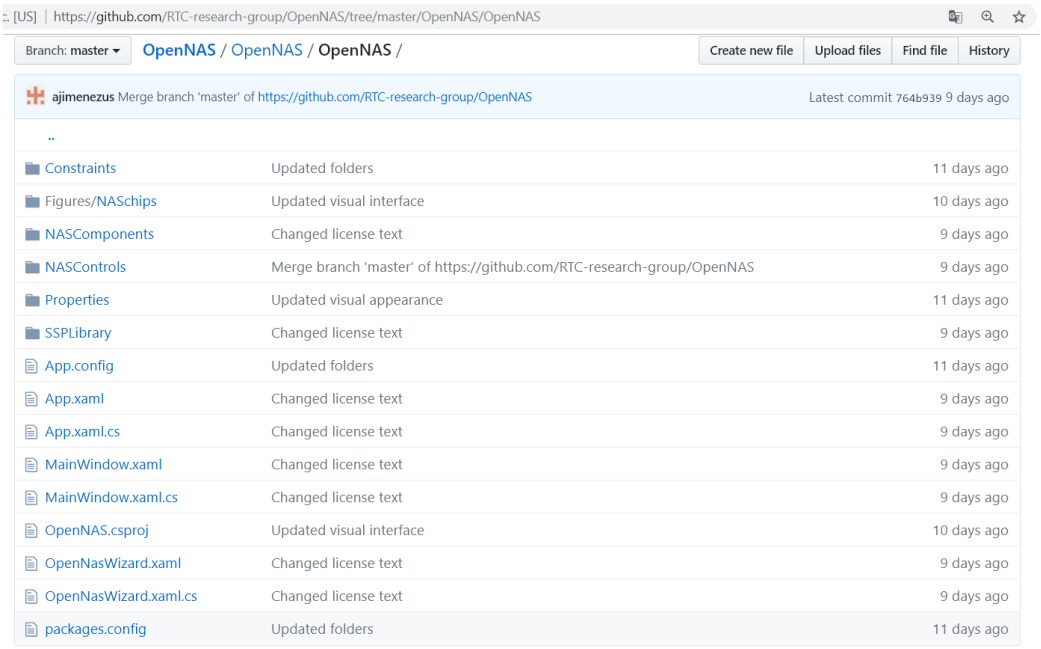

Figure 13: OpenNAS main folders.

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. The constraints file can be also generated by the tool, which will be already configured according to the options and parameters selected by the user in the wizard.
Take care with that! This file is the main error source.

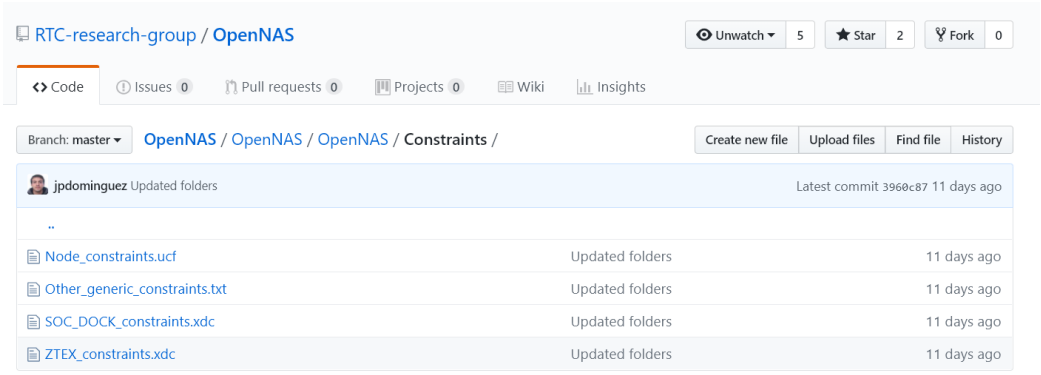

Figure 14: Constraints files available from OpenNAS.

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

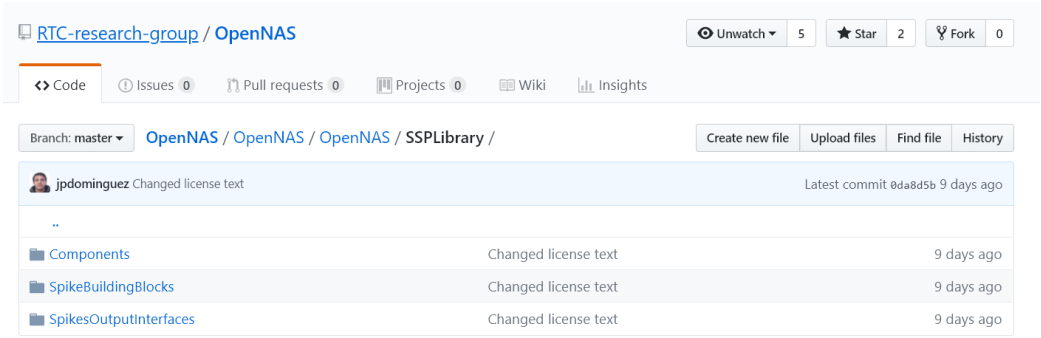

Figure 15: SSP folder contains all the NAS VHDL files.

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

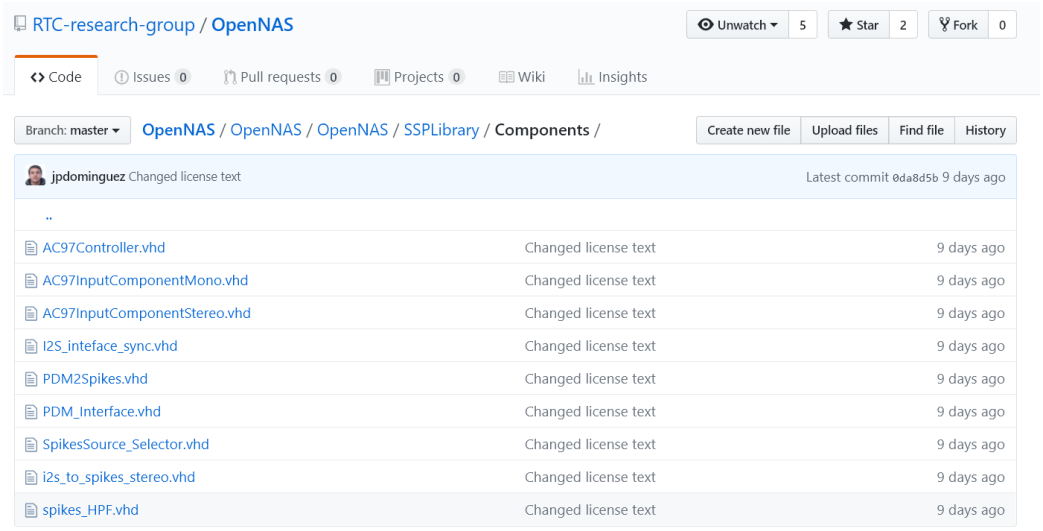

Figure 16: NAS input VHDL modules.

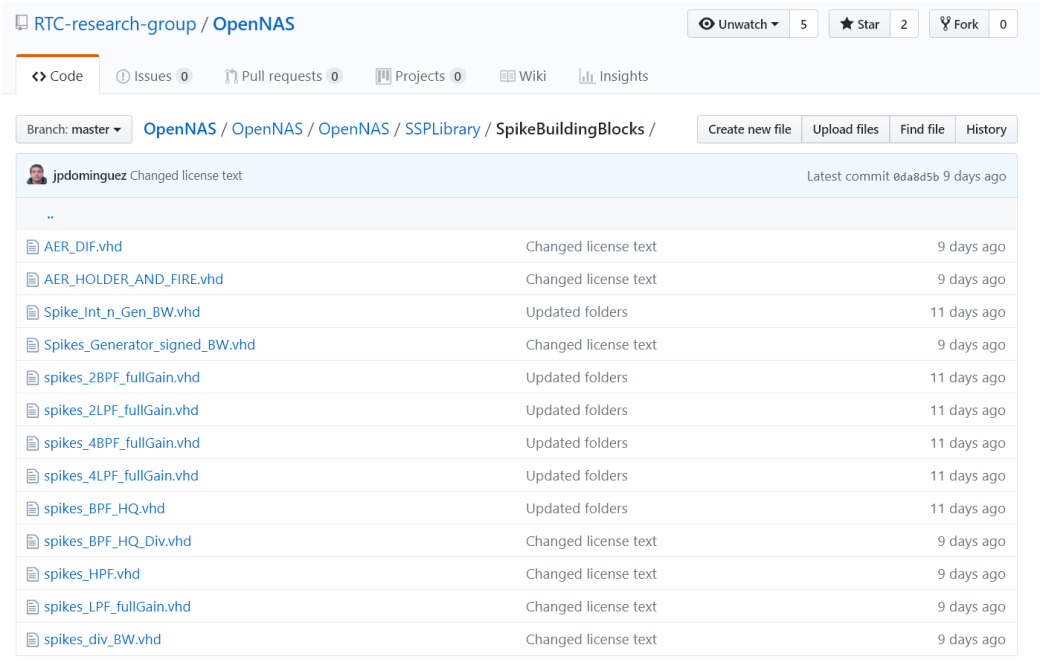

Figure 17: NAS spike-based signal processing VHDL modules.

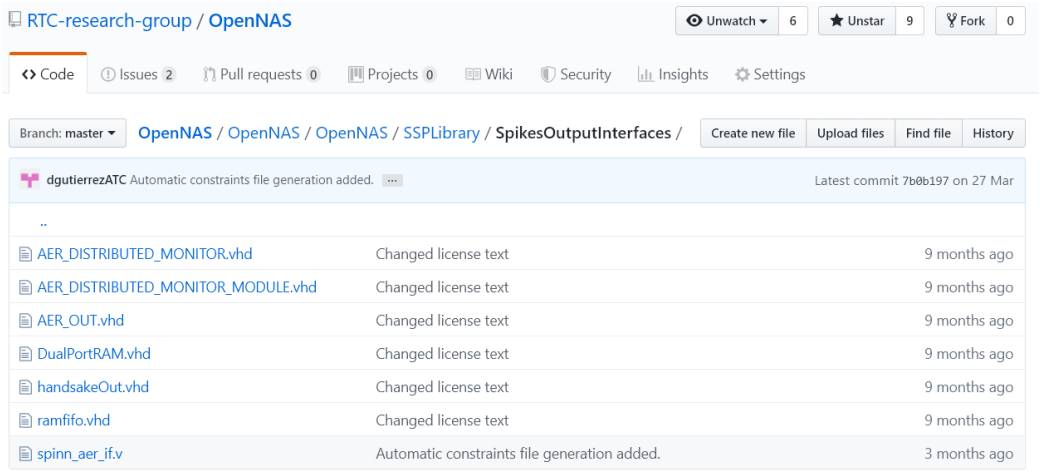

Figure 18: NAS output interfaces VHDL modules.

3.2 Software Architecture

To represent each of the NAS' components we have used a set of classes, where each class contents all the parameters and HDL information of an specific component. The class hierarchy is presented in Fig.19. The main NAS class is "OpenNasArchitecture", and it contains an instance of common parameters (OpenNASComponents) and one attribute for each of the three NAS components: AudioInput, AudioProcesingArchitecture and SpikesOutputInterface (Fig.19 mid). These are abstract classes than inherit from "HDLGenerable" abstract class (Fig.19 top), which contains the generic methods for HDL generation. Finally, specific components classes inherit from AudioInput, AudioProcesingArchitecture and SpikesOutputInterface, implementing each component features. Using this inheritance tree, a NAS is fully modeled and structured, being able to increase NAS components diversity easily and making software maintenance less tedious.

For user guidance, OpenNAS implements a graphical wizard written in WPF. User can select a complete NAS setup in five steps. We have added graphical diagrams that change interactively which each selection, for user helping and to know will be added to the final NAS design. This wizard will add specific instances to abstract members in OpenNasArchitecture class, and will fill components parameters as user selects them.

Once a full NAS design has been specified by the user, every member in OpenNasArchitecture has a specific instance and parameter set, the last step is the HDL generation. This task performs the following steps:

1. Each component writes its HDL dependency files and top entity to an output destination folder.
2. OpenNasArchitecture creates top NAS HDL file.
3. Sequentially each component writes I/O signals to NAS top file.
4. Interface signals between components are added in NAS top file.
5. Sequentially each component writes its top component architecture.
6. Inside NAS top file each component adds an invocation to its instance, and these are connected to each others using interface signals.
7. A template for constraint file is generated with all NAS I/O signals.
8. Finally, a NAS summary is written as a XML file.

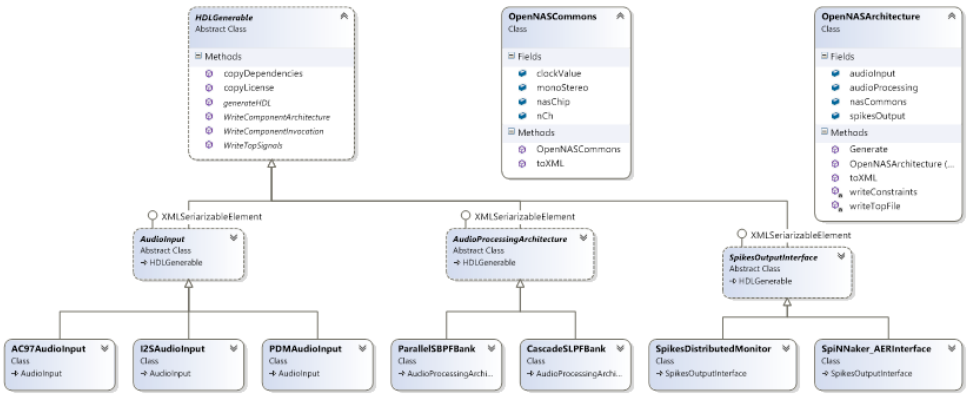

Figure 19: OpenNAS software application class diagram.

3.3 Running the VS application

It is time to run the OpenNAS application! The user can choose between either to download or to clone the repository.

3.3.1 Getting the project from GitHub

If you selected to download the project, the first step is to unzip the project. However, if you cloned the repository, you don't need to do it. Anyway, if we open the OpenNAS project folder, you will find these folders:

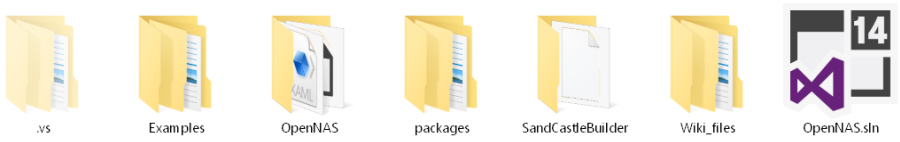

Figure 20: OpenNAS C# project files.

Then, the user can select between two options:

1. Run OpenNAS: skip the Visual Studio project steps and move to the section 3.3.3.
2. Contribute to/modify the OpenNAS project: go to the next section.

3.3.2 Launching the VisualStudio project

First, double click on the OpenNAS solution (OpenNAS.sln), and then the VisualStudio IDE 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.

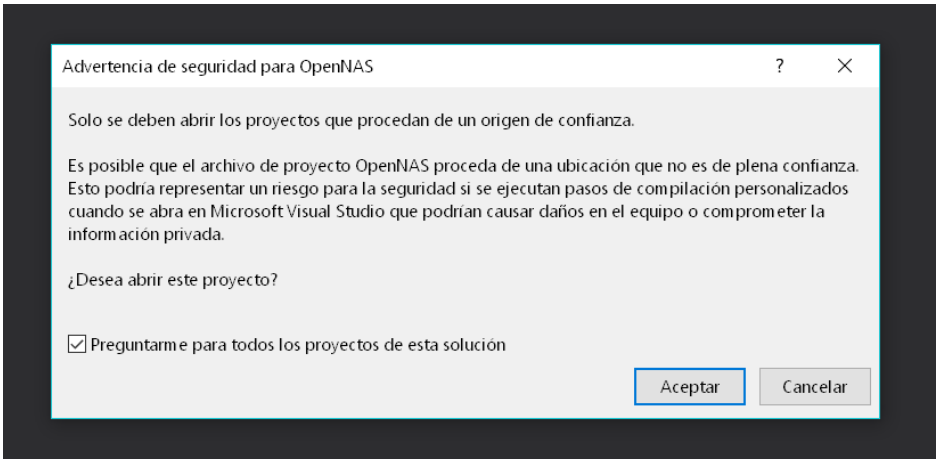

Figure 21: Advertisement launched by the VisualStudio software the first time you open the OpenNAS project.

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.

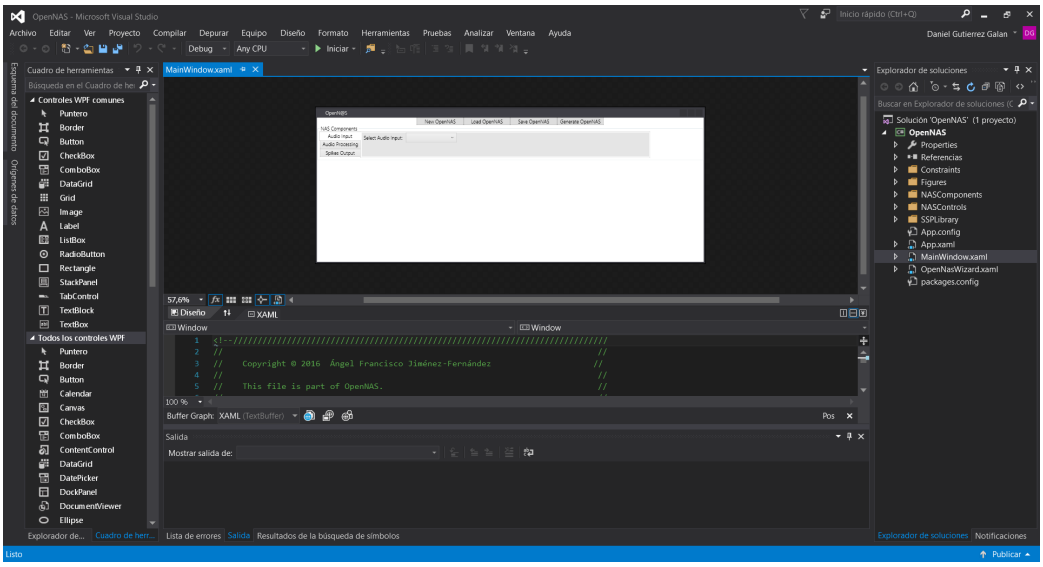

Figure 22: General view of the VisualStudio interface after open the OpenNAS project.

3.3.3 Running the OpenNAS application

For running the OpenNAS application, the required action for running the application depends on if you opened the Visual Studio Solution or not.

- For user who opened the Visual Studio Solution, click on the "Start" button, and the application should begin running.
- For users who didn't open the Visual Studio Solution: go to the OpenNAS folder → bin → Release, and double click on OpenNAS.exe file.

In both cases, a new window like the next one must appear:

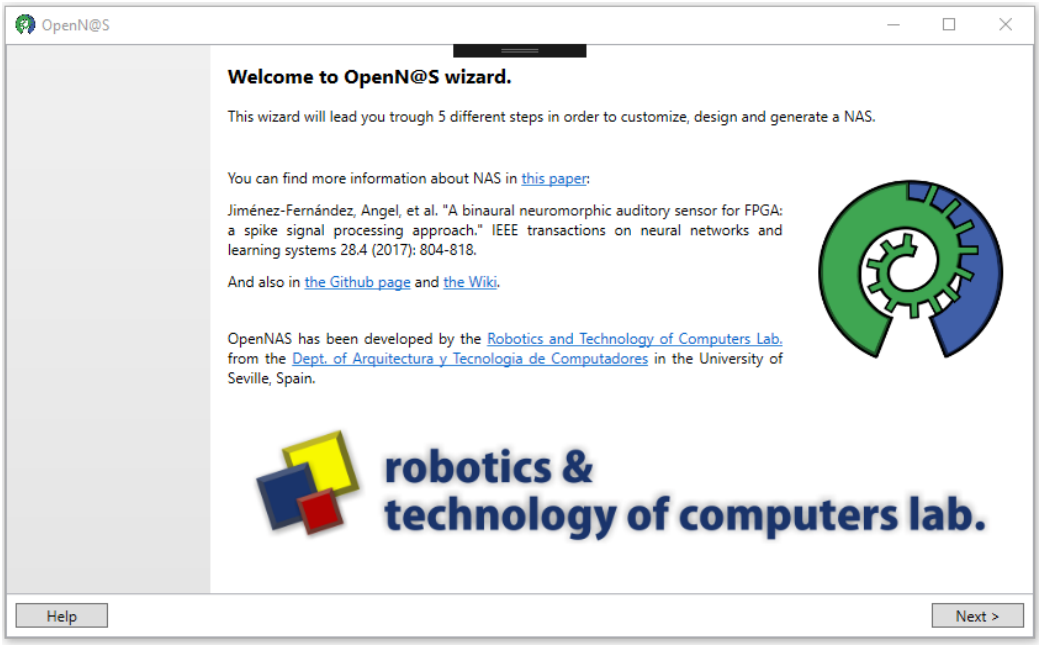

Figure 23: OpenNAS wizard welcome screen.

CONGRATULATIONS!, OpenNAS is now running!

That is the welcome screen of the wizard. Here you can find information about the original paper, the GitHub page and the OpenNAs' wiki, and a link to our University website. Click on "Next" button to start the NAS configuration.

In 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 to 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 24: First step of OpenNAS wizard. In this step, NAS common settings can be changed.

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

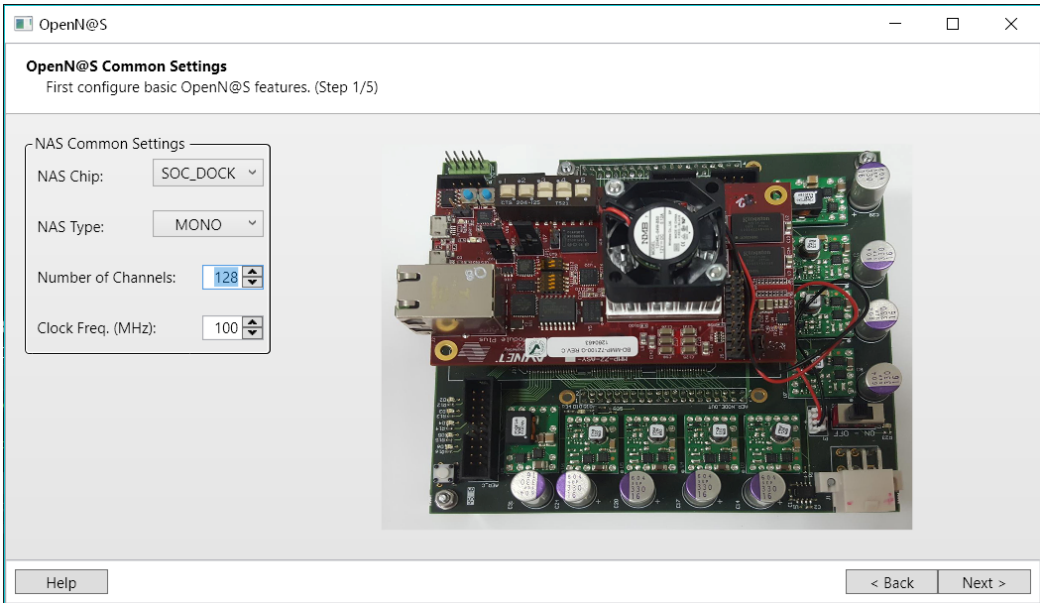

Figure 25: Setting the NAS chip to SOC_DOCK board.

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

The 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.

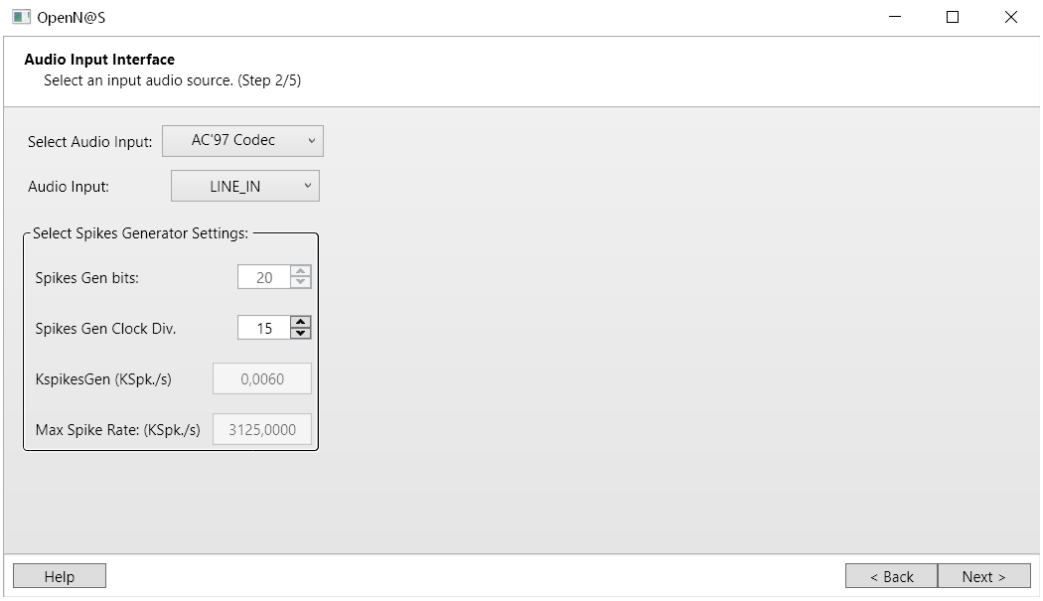

Figure 26: Audio input from AC'97 audio codec.

There exist a new version of the audio input interface. It consists of 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 uses that communication protocol to transfer the data.

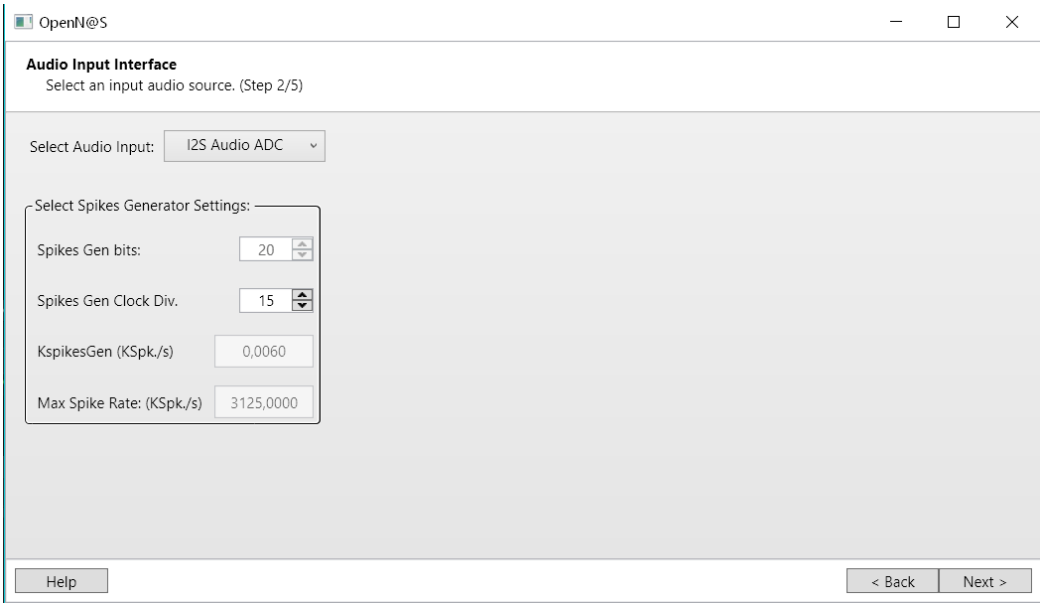

Figure 27: Audio input from line-in connector.

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

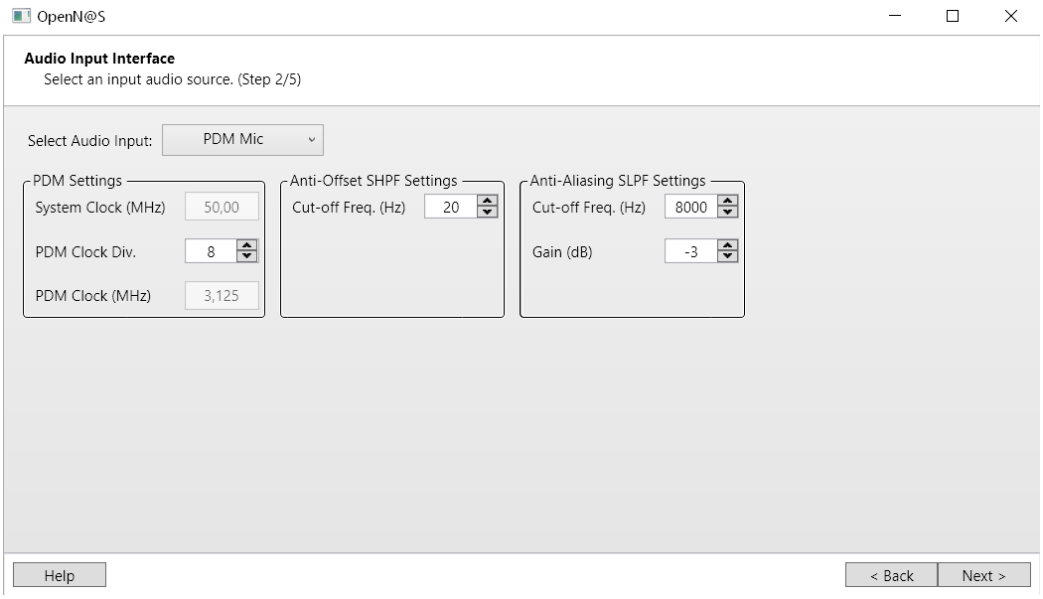

Figure 28: Audio input from PDM microphones.

Since the audio input interface board implements two different sound input devices, 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.

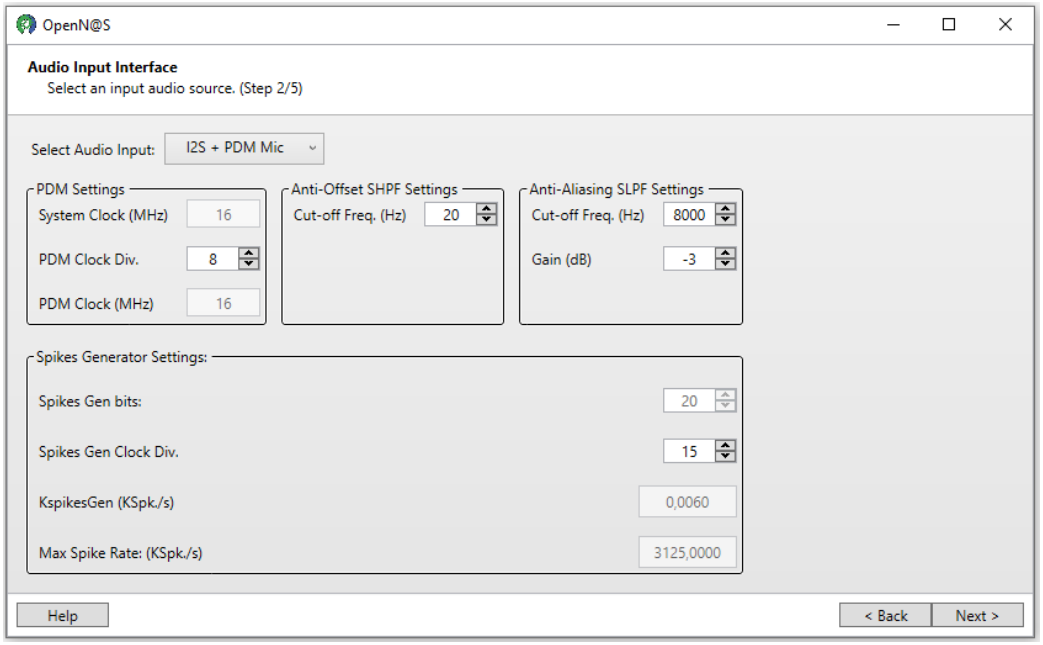

Figure 29: Audio input from either line-in connector or PDM microphones, selected by a jumper position.

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. Then, this option is selected by default. You 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 number 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.

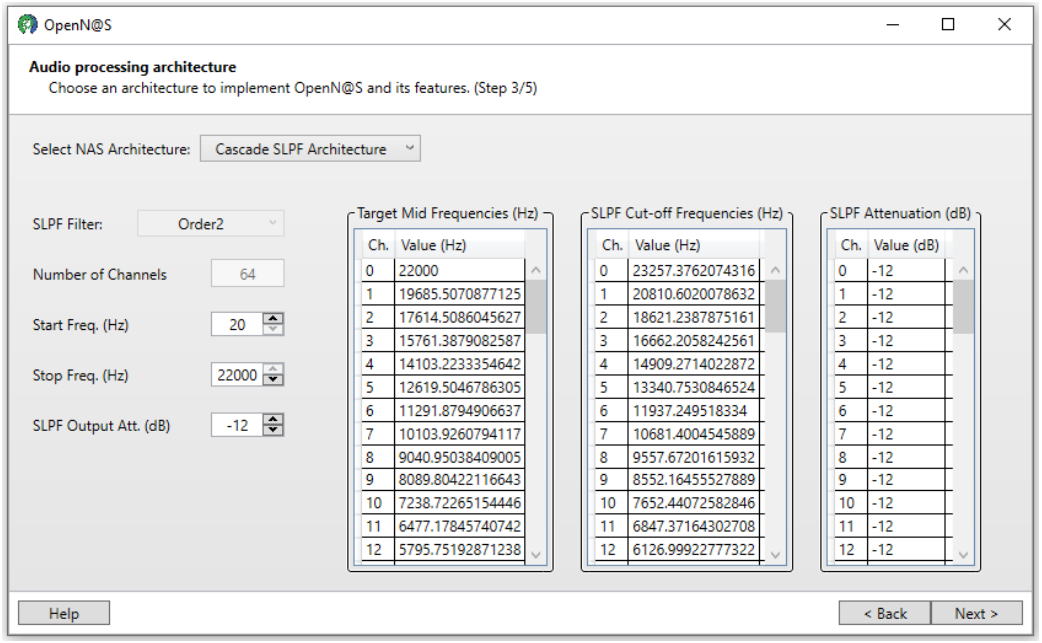

Figure 30: Selecting the NAS architecture. Cascade architecture is mostly used, and it is by default in the OpenNAS.

Filter's channels frequencies are calculated automatically by the software based on the specified architecture by the user. If you modify the attenuation value or any other parameter, all the parameters will be recalculated automatically.

There is another NAS architecture approach, the parallel filter bank architecture:

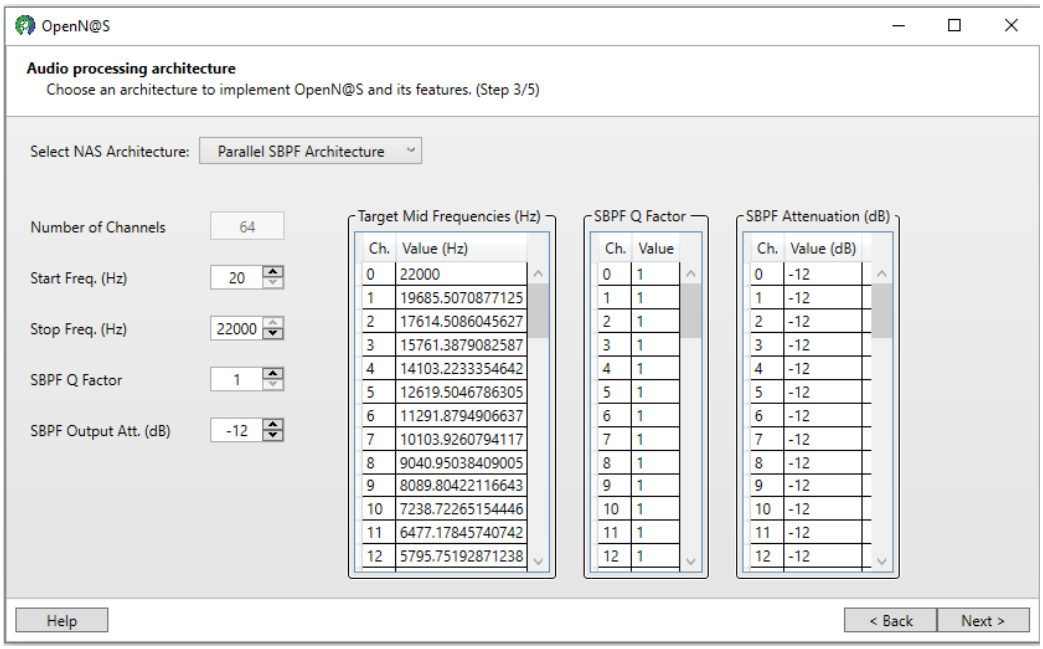

Figure 31: Parallel architecture open a different approach to the audio processing.

Almost the end, but not less important. It is time to choose the output interface. This step is completely dependent on the project in which the NAS will be used. According to 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**.

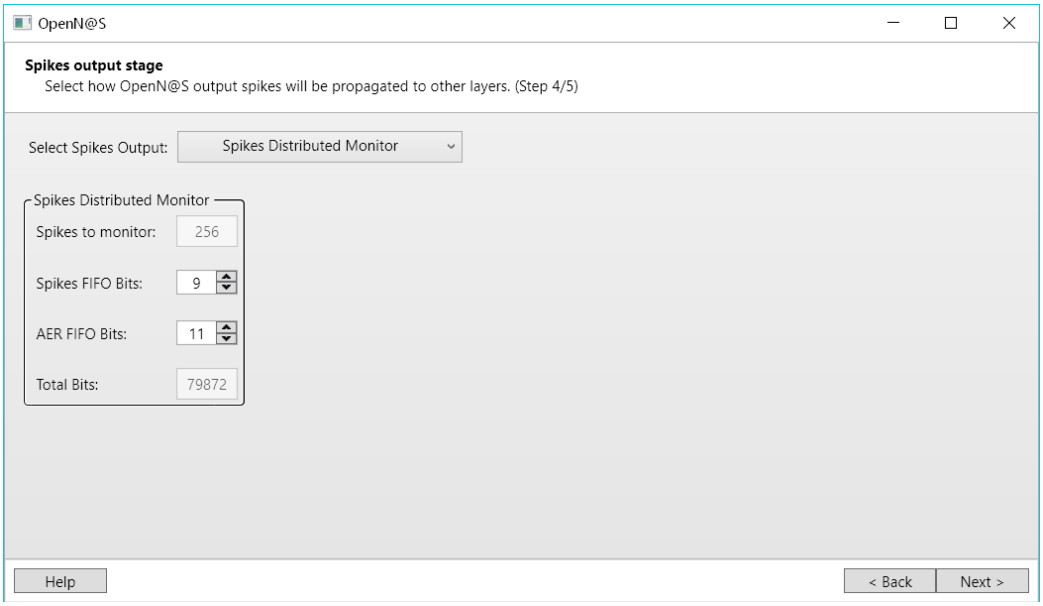

Figure 32: Using the AER Distributed Monitor as output interface allow you to monitor the NAS output in real time..

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 SpiNN3 and SpiNN5 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.

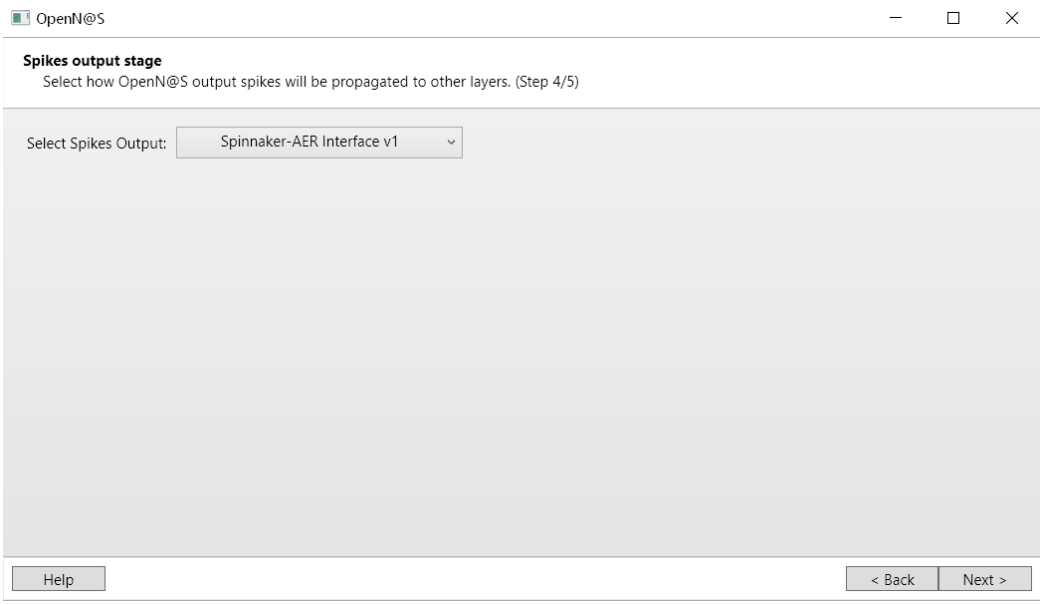

Figure 33: With SpiNNaker interface v1 you can send AER events from the NAS to the SpiNNaker machine.

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 do 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 is being sent 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 more slowly.

The last step after setting up the OpenNAS' parameters is to generate the NAS files according to the selected parameters. You need to specify a folder in which all the files will be generated.

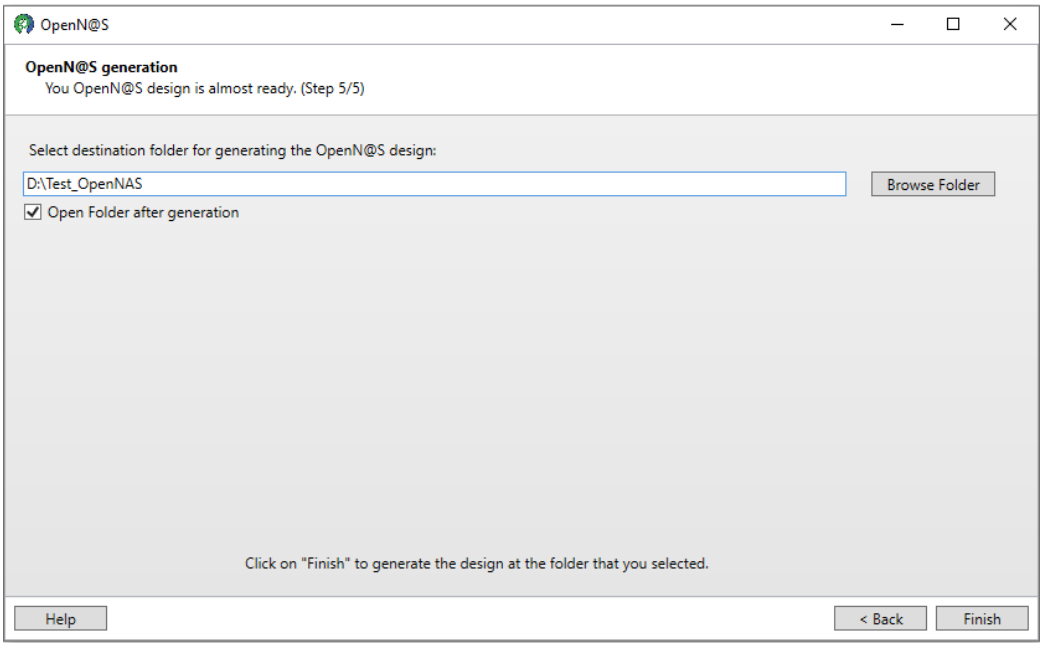

Figure 34: Select a destination folder to save the NAS files.

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 have been generated and you can take a look of the source code.

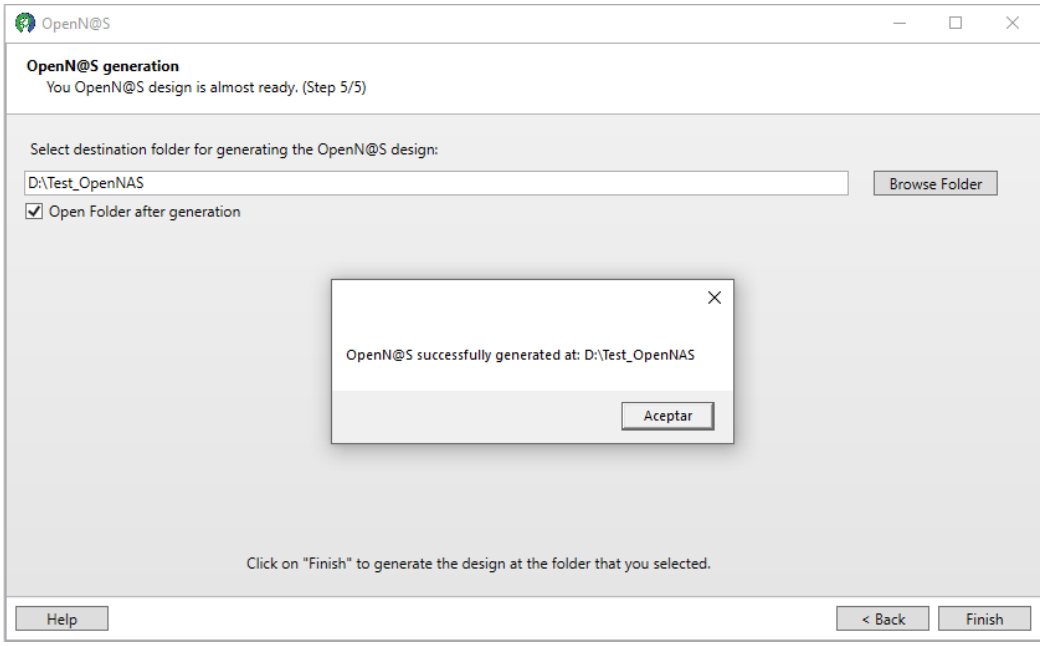

Figure 35: Files were generated successfully.

3.4 What's next? Generating the bitfile

By now, a lot of things have been taking into account to generate correctly your NAS model. And at 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 VHDL project to, finally, generate the programming file with .bit extension and then 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.

If not, if you do not have one of those boards or you would like to modify the source files, you will need to create an ISE project manually, to configure it correctly and to add the source files by hand.

3.4.1 Creating a Xilinx ISE project

The first step is to open the ISE 32-bit Project Navigator. The icon looks like is showed in the next Figure 36:

Figure 36: ISE Project Navigator icon.

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

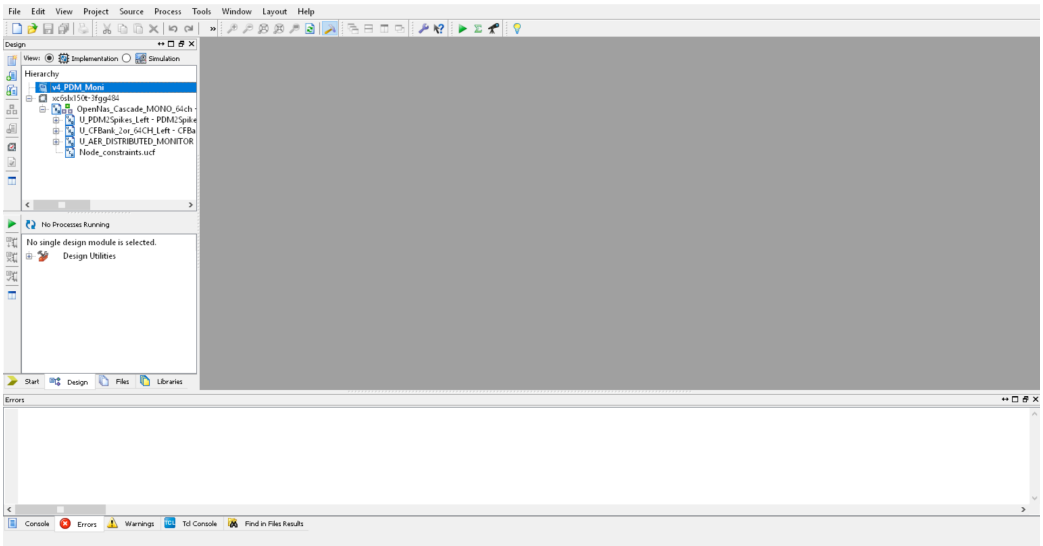

Figure 37: ISE Project Navigator main view.

We need first to create a new project. To do that, go to File, and select New Project.

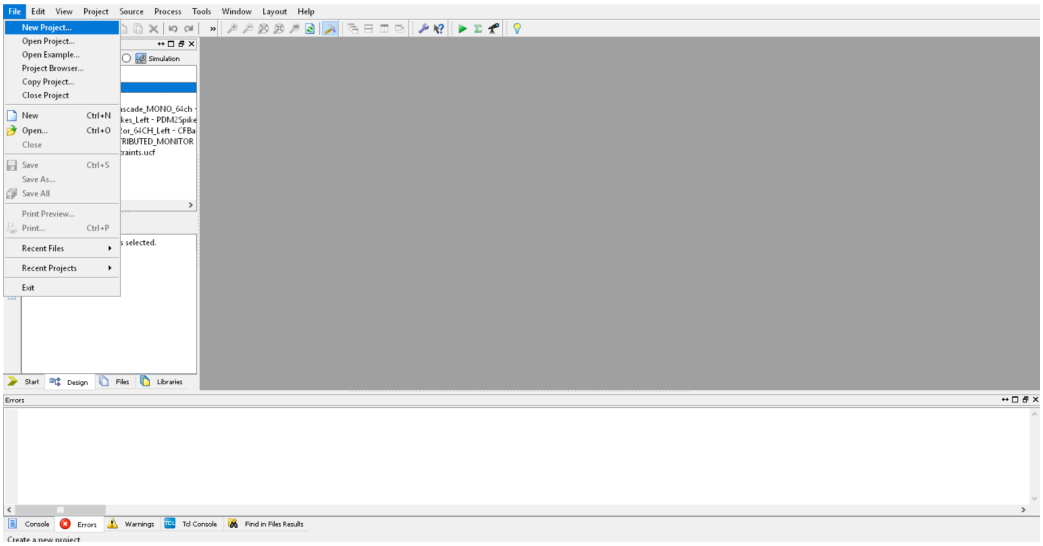

Figure 38: Creating a new project in ISE Project Navigator.

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

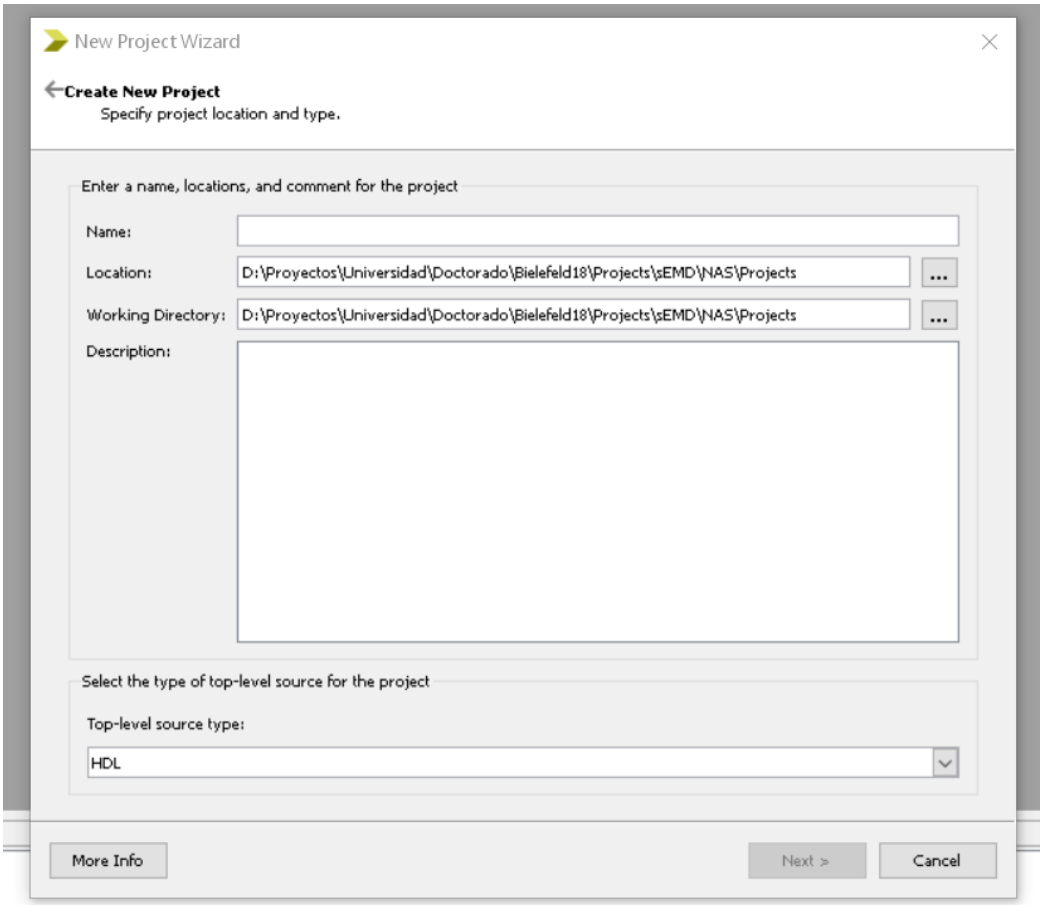

Figure 39: ISE new project wizard.

Like in every new project, you have to indicate the project name. According to the folders architecture aforementioned, we recommend selecting 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.

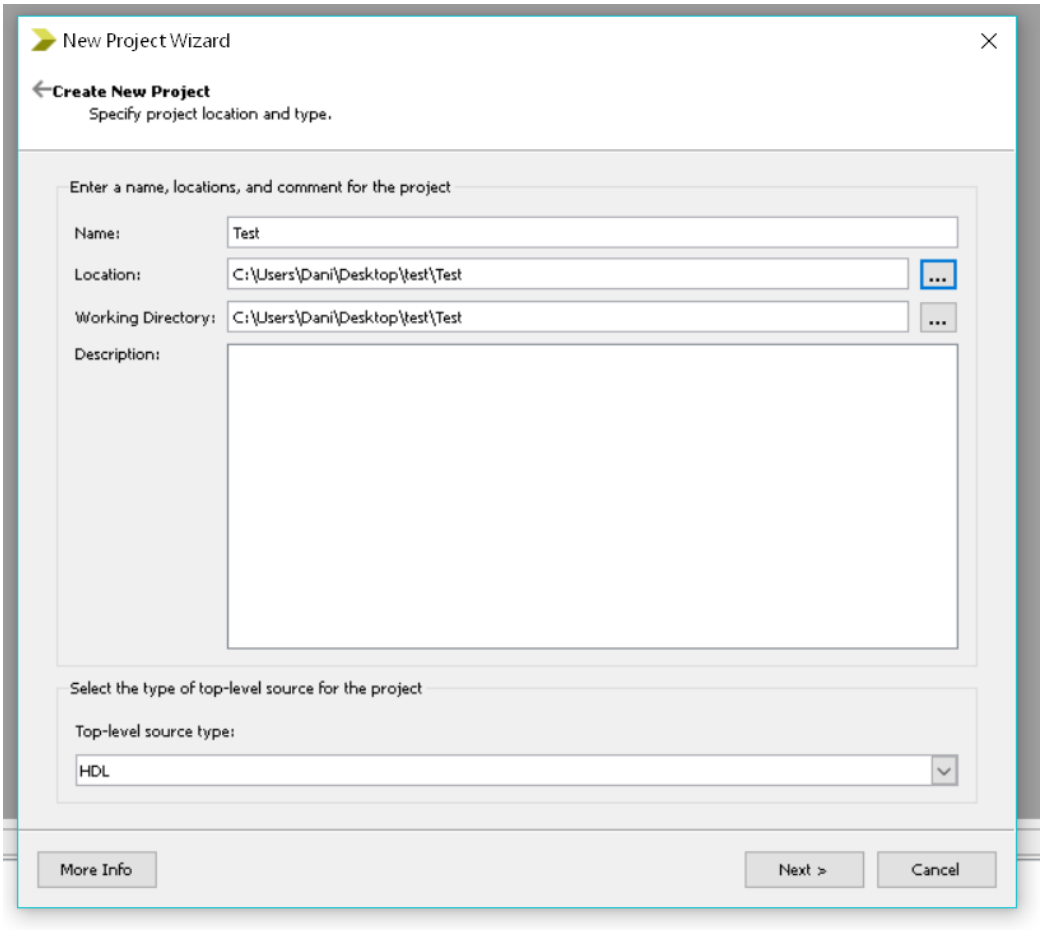

Figure 40: Setting the ISE project name and the project localization.

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, the user should make its own project settings.

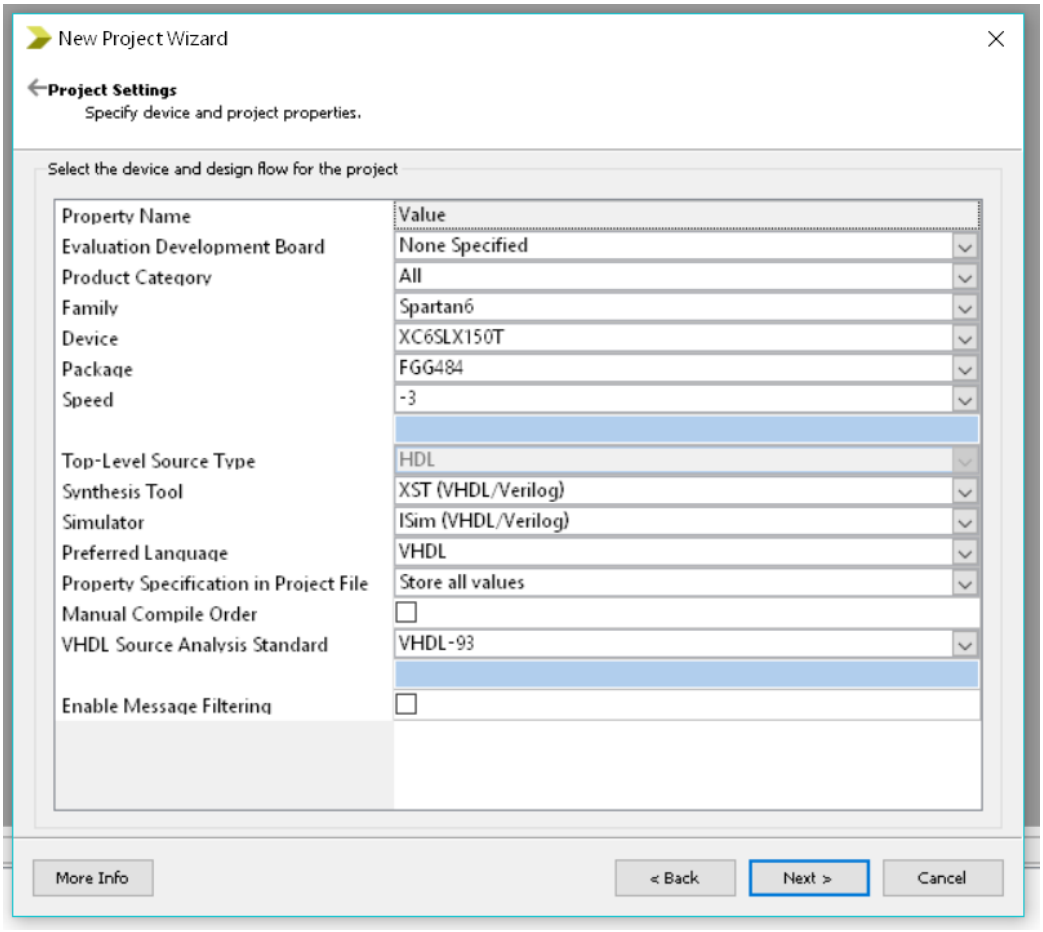

Figure 41: Correct project settings to create a ISE project for the AER-Node board.

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

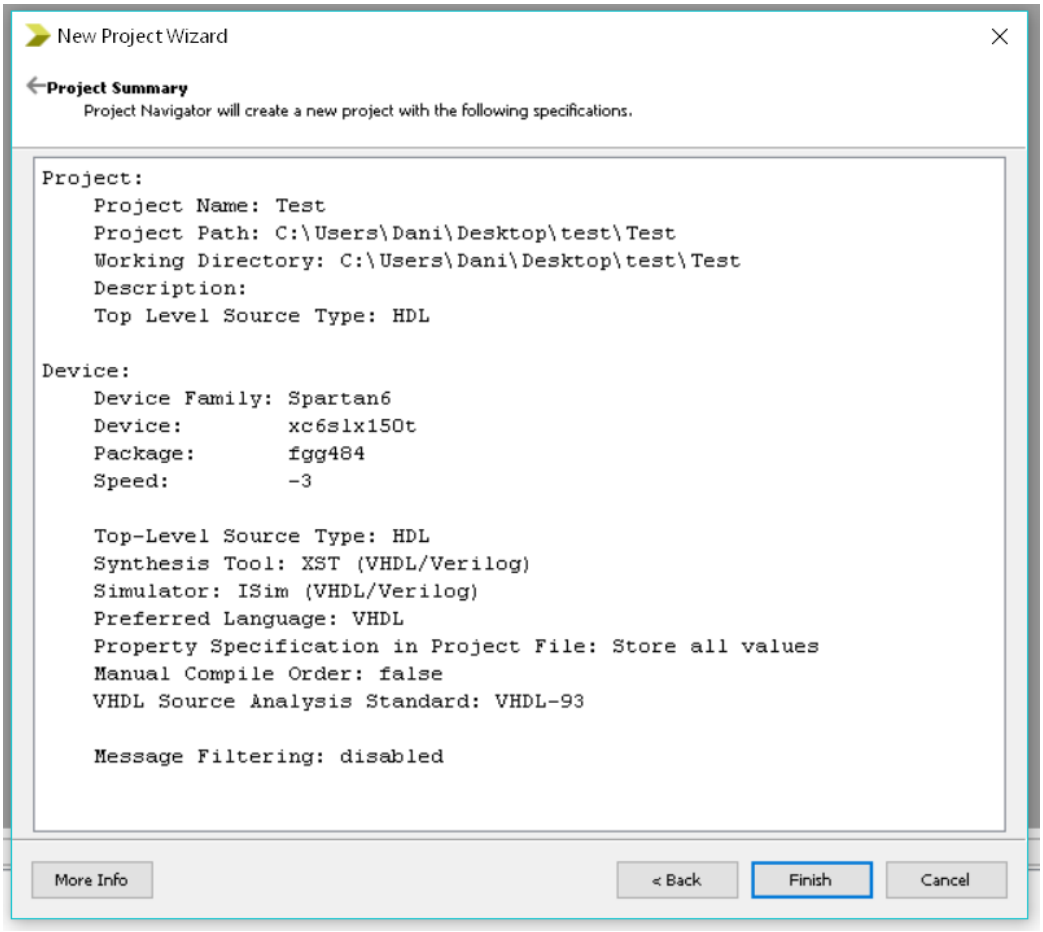

Figure 42: ISE project settings summary.

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.

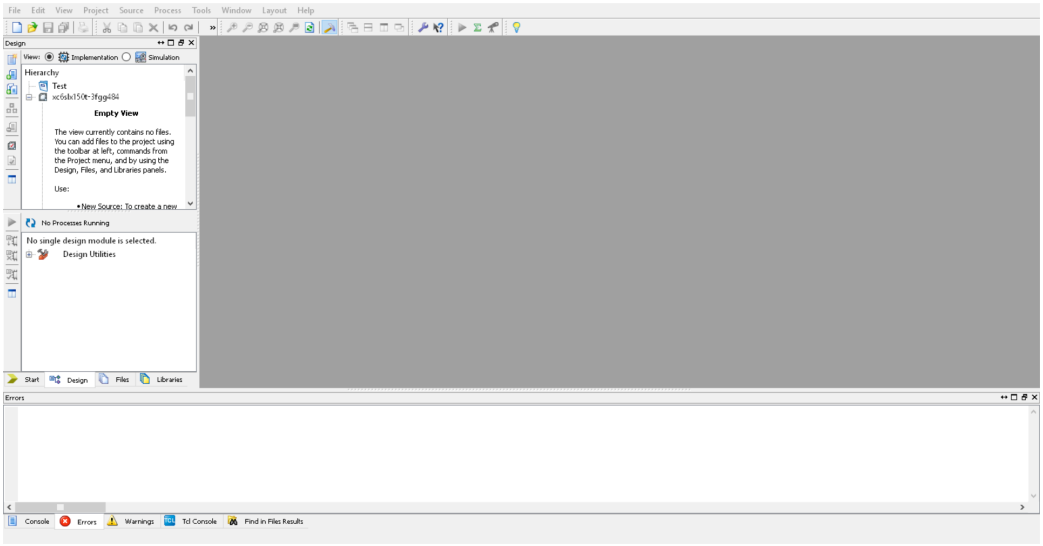

Figure 43: Main view after create a new project.

3.4.2 Adding the NAS source files to the project

The NAS VHDL files generated by the OpenNAS will be now added into the new ISE project. First, make right click either on the project folder or on the selected chip. A new menu appears. Then select "Add sources".

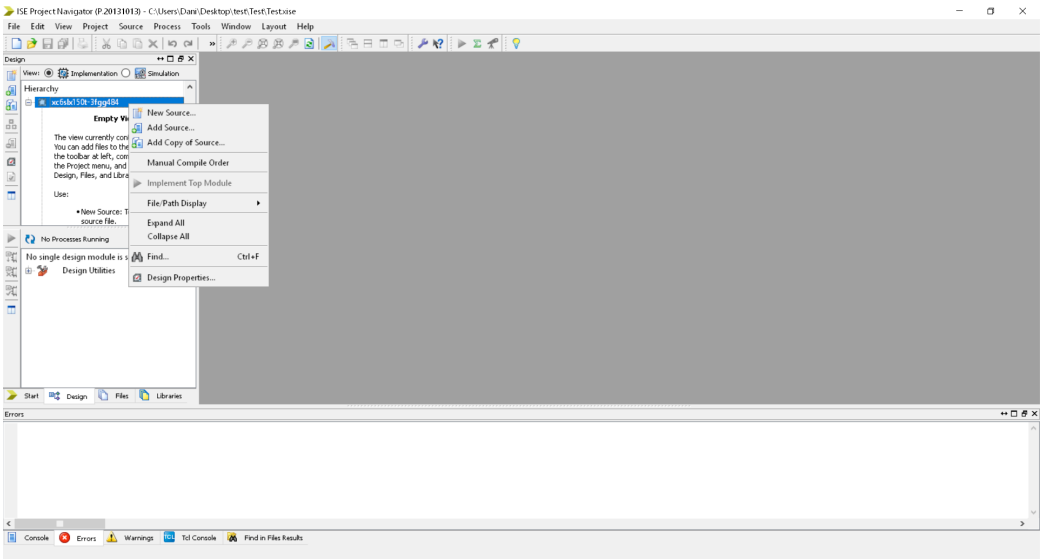

Figure 44: Ready for adding the NAS VHDL sources.

An emergent window 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. The source files should be in ../NAS_PDM_64ch_MONO_Monitor/sources/.

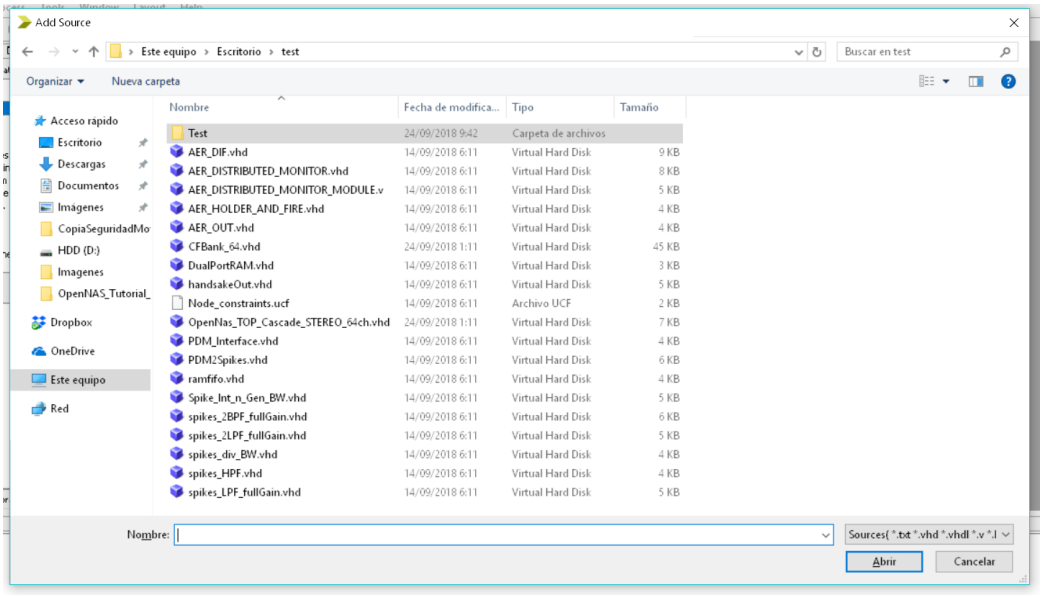

Figure 45: Go to the destination folder in which the NAS VHDL were saved.

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

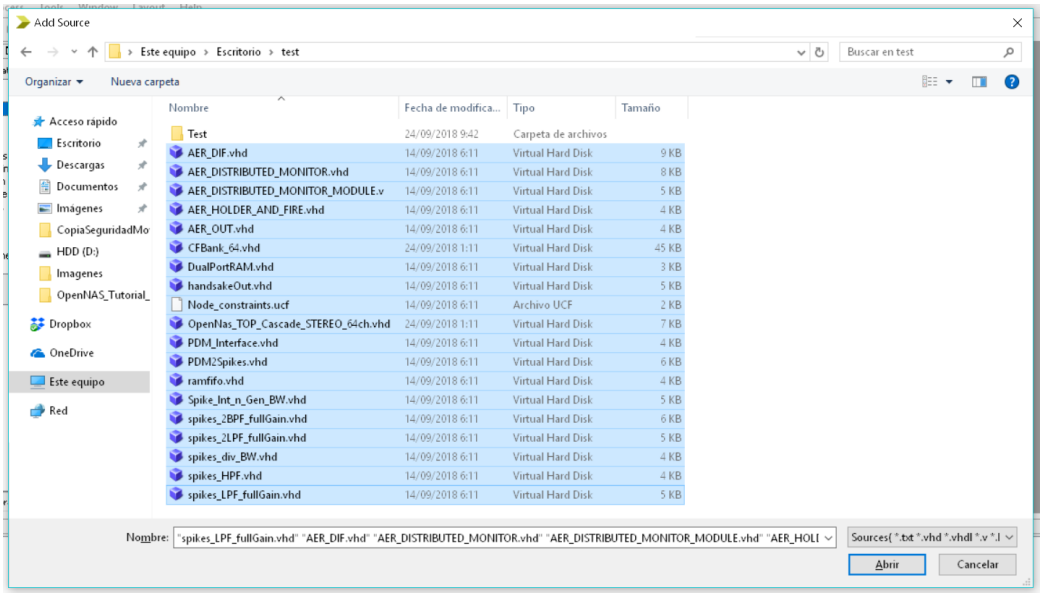

Figure 46: Select all the files, except the .xml file.

In an 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 to 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.

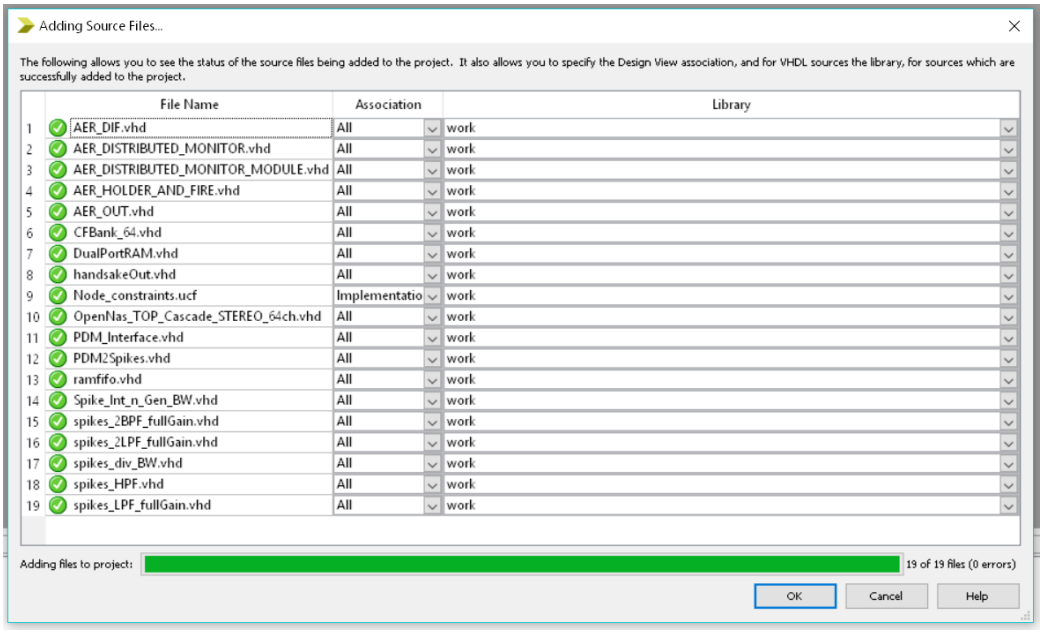

Figure 47: Selecting the files association.

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

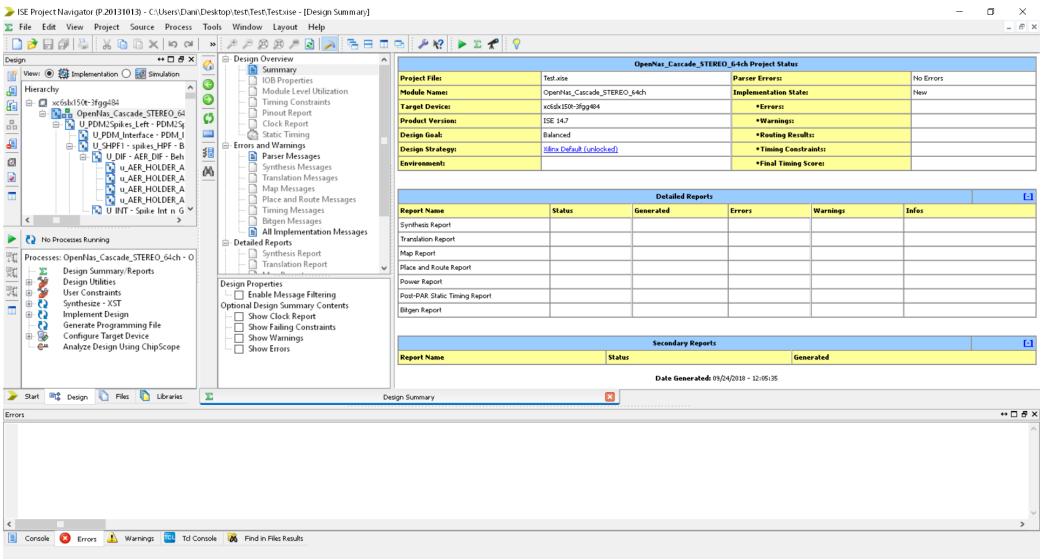

Figure 48: ISE main window after to add the source files.

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).

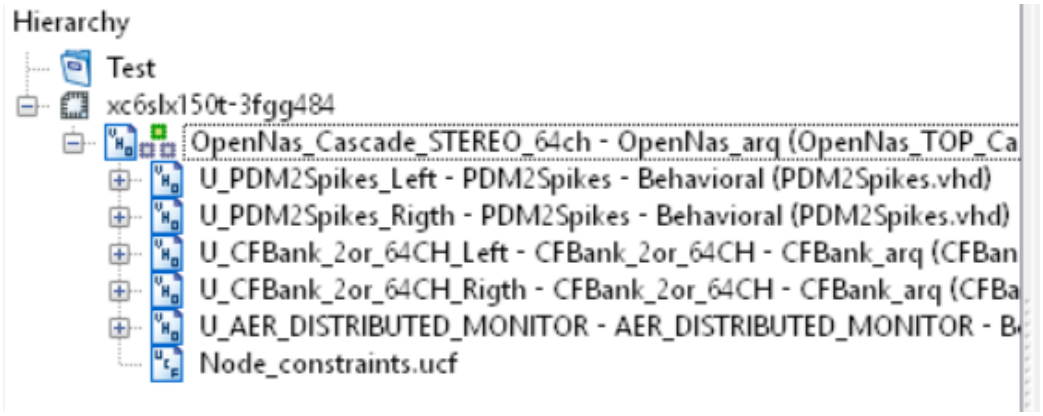

Figure 49: OpenNAS project modules hierarchy.

3.4.3 Generating the programming file

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

options will appear.

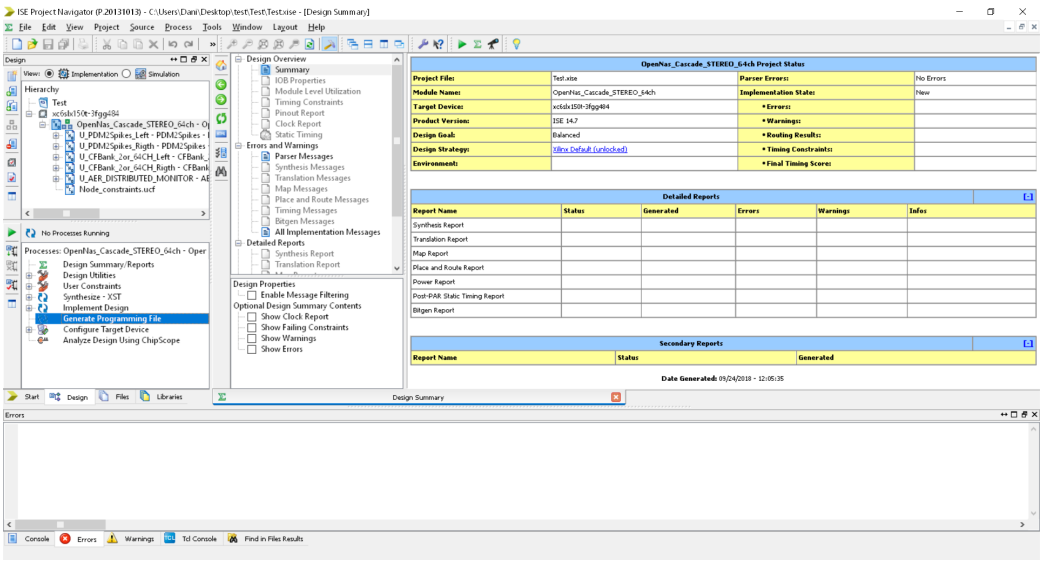

Figure 50: Clicking on the top file to show the process menu.

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.

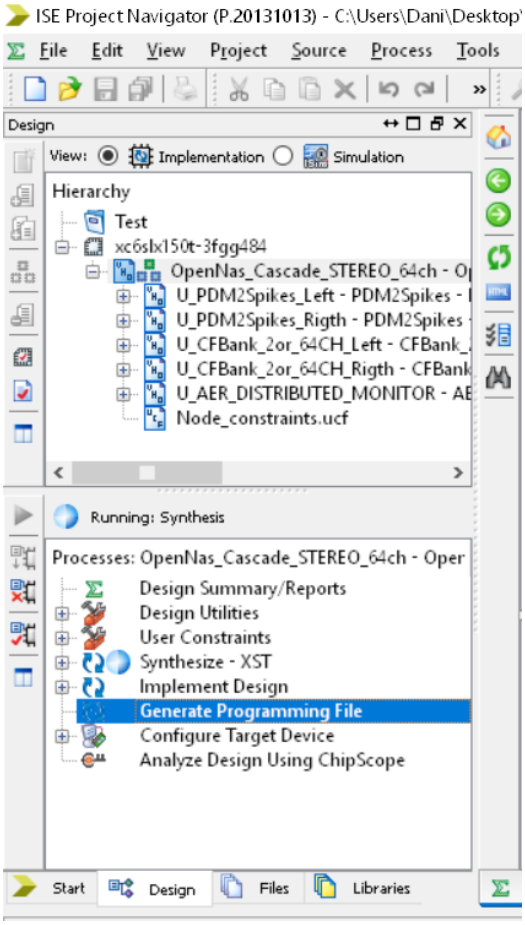

Figure 51: Running the Generation Programming File process.

This process will take between 15 and 20 minutes depending on your computer's resources.

3.4.4 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. This process is for being sure that the communication between both boards is synchronized.

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.