Department of Information Engineering(DII)
Master of Science in Computer Engineering

IIR Filter For Audio Application

Arsalen Bidani
And
Tesfaye Yimam Mohammad

Semester Final Project of Electronics course.

Winter Session
Academic Year 2021/22

Contents

List of Figures

1 Introduction

1.1 Digital Filters

Filters are essential building blocks of any **Electronic** and **Communication Systems** that alter the amplitude and/or phase characteristics of a signal with respect to frequency. Filter is basically linear circuit that helps to remove unwanted components such as Noise, Interference and Distortion from the input signal. i.e the filter selects from a frequency or range of frequencies out of a mix of different frequencies from the input.

From the domain of implementation filters can be categorized as analog and/or digital ones.The analogs were mathematically modeled using ordinary differential equations of Laplace transforms. They were analyzed in the time or **s** ,also known as Laplace, domain.A digital filter is a mathematical algorithm implemented in hardware and/or software that operates on a digital input signal to produce a digital output signal for the purpose of achieving a filtering objective.

According to [2] , digital filters are basically used to modify or alter the attributes of a signal in the time or frequency domain.Digital filters may be more expensive than an equivalent analog filters due to their increased complexity, but they make practical many designs that are impractical or impossible as analog filters. In digital filters, the coefficient values are stored in computer memory, making them far more stable and predictable.

Hence,the motivation for studying digital filters is found in their growing popularity as a primary DSP operation. Digital filters are rapidly replacing classic analog filters, which were implemented using RLC components and operational amplifiers.Depending on whether the duration of the impulse response is finite or infinite, digital filters are classified as **FIR** or **IIR** filters. The former is mathematically expalined by equation (1) and the later, because of its infinite impulse response, has the expression in (2).

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \tag{1}$$

FIR filters	IIR Filters
Can have exactly linear phase response, i.e. do not distort the phase of the signal.	The phase responses are nonlinear, especially at the band edges.
FIR filters realized non-recursively are always stable.	The stability of IIR filters cannot be guaranteed.
The effects of using limited number of bits to implement filters are less severe.	The effects like roundoff noise and coefficient quantization errors are more severe
Require more coefficients for sharp cutoff filters.	Less coefficients and thus less processing time and storage.
No analog counterpart, but, easier to synthesize filters with arbitrary frequency responses.	Analog filters can be readily transformed into equivalent IIR digital filters.
Algebraically more difficult to synthesize if CAD support is not available.	Less difficult to synthesize.

Table 1: Brief Summary of FIR and IIR filters

$$y[n] = \sum_{k=0}^{\infty} \mathbf{h}[k]\mathbf{x}[n-k] = \sum_{k=0}^{N} b_k \mathbf{x}[n-k] - \sum_{k=1}^{M} a_k \mathbf{y}[n-k] \tag{2}$$

A digital filter in general can be represented by its impulse response,h[k], as

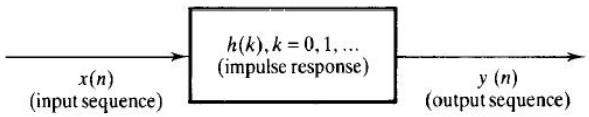


Figure 1: Conceptual Representation of a Digital Filter [1]

depicted in fig.1.Summary of FIR and IIR filters is given in Table 1.

It is common in DSP to say that a filter's input and output signals are in the time domain. This is because signals are usually created by sampling at regular intervals of time. Several time-domain representations for digital filter exists such as the difference equations and the impulse response , transfer

function and frequency response. Each of these responses contains complete information about the filter, but in a different form. If one of the three is specified, the other two are fixed and can be directly calculated. All three of these representations are important, because they describe how the filter will react under different circumstances.

1.1.1 Difference Equation

Digital filters are characterized by difference equations of the form 2.
In the input-output difference equation above, the output y[n] is given as the linear combination of present and past inputs minus a linear combination of past outputs (feedback term). The parameters a_i and b_i are the filter coefficients or filter taps and they control the frequency response characteristics of the digital filter.Filter coefficients are programmable and can be made adaptive (time-varying).A direct-form realization of the digital filter is shown in fig.6 The filter in Eq. 2 is referred to as an infinite-length impulse response (IIR) filter. The impulse response, h(n), of the filter shown in Fig. 6 is given by eq. 3

$$h(n) = \sum_{i=0}^{L} b_i \delta(n-i) - \sum_{i=0}^{M} a_i h(n-i)) \tag{3}$$

The IIR classification stems from the fact that, when the feedback coefficients are non-zero, the impulse response is infinitely long.

1.1.2 Impulse Response

In addition to difference-equation coefficients, any LTI filter may be represented in the time domain by its response to a specific signal called the impulse. This response is called, naturally enough, the impulse response of the filter.

By definition the impulse signal is denoted as $\delta(n)$ and defined by
$\delta(n) = \begin{cases} 1, \ n = 0 \\ 0, \ n \neq 0. \end{cases}$ The impulse response an example second-order system is shown in fig.2

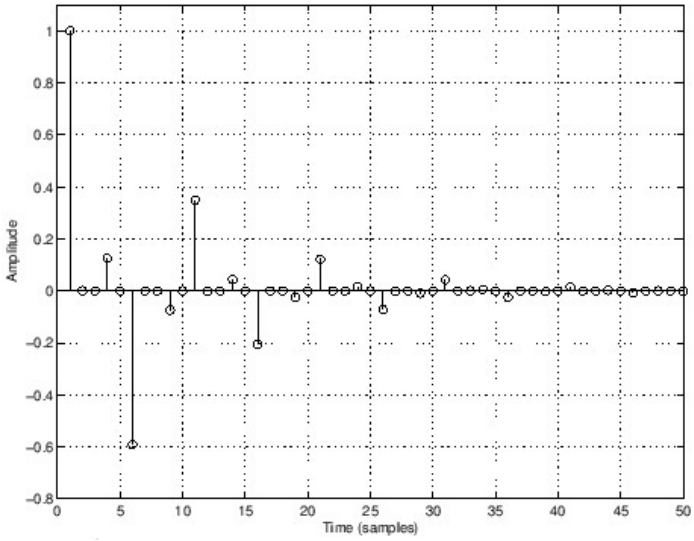


Figure 2: Impulse response of a second-order system

1.1.3 The Transfer and Frequency Response Functions

The z-transform of the impulse response of a filter is called the transfer function and is given by 4.

$$H(z) = \sum_{n=-\infty}^{+\infty} h(n) z^{-n} \tag{4}$$

By collecting the mathematically like terms from eq. 2 and taking the ratio of the output over the input one can get the formula at 5 which is the transfer function in terms of the coefficients of the filter.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_L z^{-L}}{1 + a_1 z^{-1} + ... + a_M z^{-M}} \tag{5}$$

The frequency response function is a special case of the transfer function of the filter. As a result it can be easily derived from eq.5 More detail is available in [3].

1.1.4 Stability Condition

A z domain function can be written in terms of its poles and zeros as eq.6.

$$H(z) = G\frac{(z-\zeta_1)(z-\zeta_2)...(z-\zeta_L)}{(z-p_1)(z-p_2)...(z-p_M)} = G\frac{\prod_{i=1}^{L}(z-\zeta_i)}{\prod_{i=1}^{M}(z-p_i)} \tag{6}$$

where ζ_i and p_i are the zeros and poles of H(z), respectively, and G is a constant. The locations of the poles and zeros determine the **stability** of the filter.Generally speaking if the poles and zeros are within the *unit circle* of the the pole-zero plot it is a stable filter, on the other hand if they are out side of the unit circle the filter is said to be unstable. When those values are on the unit circle it will be problematic to say the filter is stable as 'n' goes to infinity.

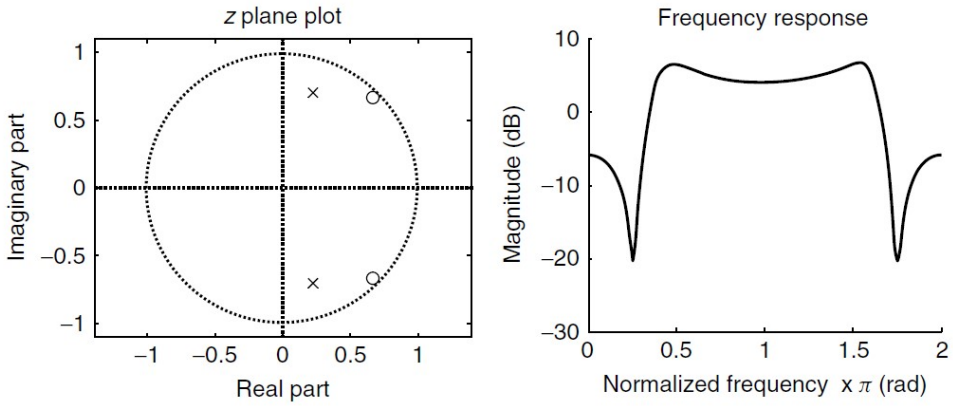


Figure 3: Pole-zero (left) and Frequency Response(right) Plots of a Second-Order System

1.2 Filter Audio Application and WAV Audio Format

1.2.1 Application Areas

Due to the versatile nature of digital filters we come across digital filters in many areas in our daily basis such as data compression,biomedical signal

processing, speech and image processing, data transmission, digital audio, telephone echo cancellation, in the music industry to mention a few.

Digital Audio processing benefits the most especially the possibility of modifying the signal spectrum as it is a very common requirement for all digital audio signal processing (DASP) applications. The device for its implementation, generally known as a filter, is almost always present in both professional and consumer equipment. For example the variation of the spectrum may be necessary for the acoustic correction of the listening environment or instead, more simply, it may be guided by the musical tastes of the listener. Applications include real-time audio effects, such as convolution reverbs, digital room equalization, audio rendering for computer games and acoustic virtual reality, spatial sound reproduction techniques, crosstalk cancellation and many more.

1.2.2 WAV Audio Format

The **WAV** file is an instance of a Resource Interchange File Format (RIFF) defined by IBM and Microsoft [4]. The RIFF format acts as a **wrapper** for various audio coding formats.Though a WAV file can contain compressed audio, the most common WAV audio format is uncompressed audio in the linear-pulse-code-modulation (LPCM) format. LPCM is also the standard audio coding format for audio CDs, which store two-channel LPCM audio sampled at 44.1kHz with 16 bits per sample. Since LPCM is uncompressed and retains all of the samples of an audio track, professional users or audio experts may use the WAV format with LPCM audio for maximum audio quality.

Unlike the mp3 file formats,WAV files are ideal for use in CD, iTunes/Google Play, TV, radio, DVD, video and other media which require really high audio quality. This particular audio format is lossless and retains the maximum audio quality.

Beginning with Windows 2000, a **wave format extensible** header was defined which specifies multiple audio channel data along with speaker positions, eliminates ambiguity regarding sample types and container sizes in the

standard WAV format and supports defining custom extensions to the format chunk.There are some inconsistencies in the WAV format: for example, 8-bit data is unsigned while 16-bit data is signed.

1.3 Project Statement

For the Electronics call winter session project we were consequently assigned the task of designing a digital circuit implementing an Infinite Impulse Response filter for audio application that follows the difference equation of (7).

$$y[n] = y[n-1] - \frac{1}{4}x[n] + \frac{1}{4}x[n-4] \tag{7}$$

As the assignment states, the following filter allows reduces the components of the input signal that have a frequency equal to half the sampling frequency. This IIR Filter operates on 16 bits of input and output in fact below is depicted the final block diagram of the circuit :

Figure 4: Diagram of the IIR Filter

1.4 Architectures of IIR Filters

While dealing with digital filters there are different signal flow diagram architectures to choose from. Computations of y[n] can be arranged in different ways to give the same difference equation, which leads to different structures for realization of discrete-time LTI systems. Basic operations in block diagram are shown in the figure 5 below

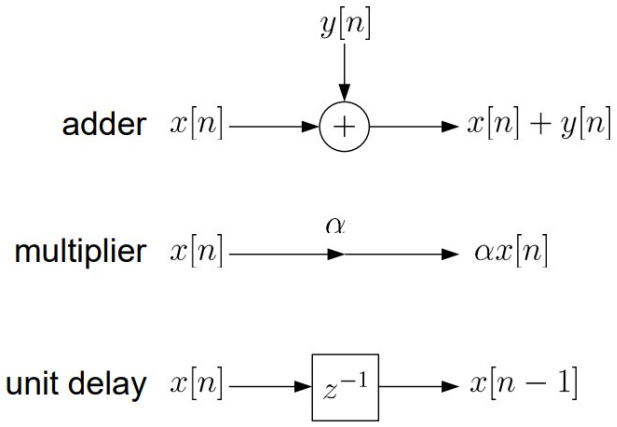


Figure 5: Basic Operations in Block Diagram

Several practical realization architectures that implement the digital IIR filter were outlined as the following :

1.4.1 Direct Form I

The digital IIR filter can be realized in the direct form I according to eq. (1) which comprises a non – recursive part and recursive part. The implementation of the digital IIR filter in this form is shown in Figure 6; for each output sample, it requires (N + M) delays, (N+M+1) multiplications, and (N+M) additions.

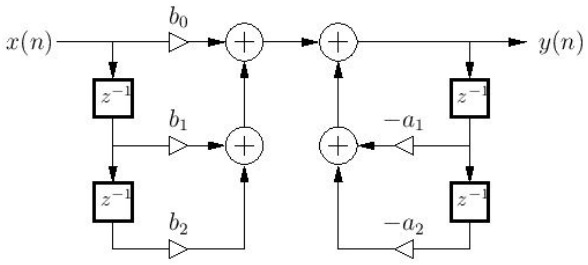


Figure 6: Filter Structure: Direct-Form-I

1.4.2 Direct Form II

In the direct form II structure, shared delay elements between the recursive and non – recursive parts of the digital IIR filter are employed, as shown in Figure 7.

The determination process of each output sample requires max (N or M) delays, (N+M) additions, and (N+M+1) multiplications.

The benefit of DFII is its more economical utilization of the delay units. However, both separate the poles sections from the zero sections, DFII can share the delay units between them. Therefore, a significant reduction in the number of delay units can be achieved when adopting DFII. The drawback of the DFII is that the pole units precede the zero units and imposes an unfeasible dynamic range on the delay units' intersection at some frequencies. As a result, DFII suffers from overflow, while DFI has immune from this effect.

1.4.3 Transposed Direct Form II

Filter transposition may also be called flow graph reversal, and transposing a Single-Input, Single-Output (SISO) filter does not alter its transfer function. The transposed structure of the DFII, shown in Figure 8, precedes the zero sections and shares the delay units between zeros and poles. Therefore, this structure has the advantage of DFII with more robust behavior. An advantage of the transposed direct-form II structure 8 is that the zeros effectively

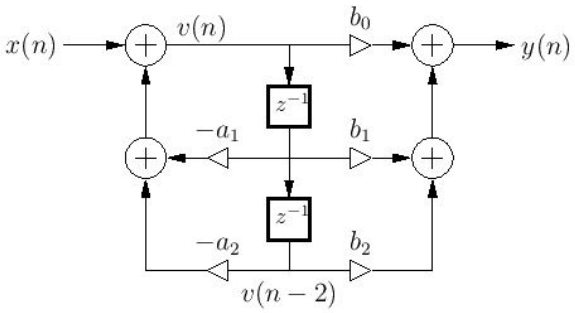


Figure 7: Filter Structure: Direct-Form-II

precede the poles in series order.

As mentioned above, in many digital filters design, the poles by themselves give a large gain at some frequencies, and the zeros often provide compensating attenuation.

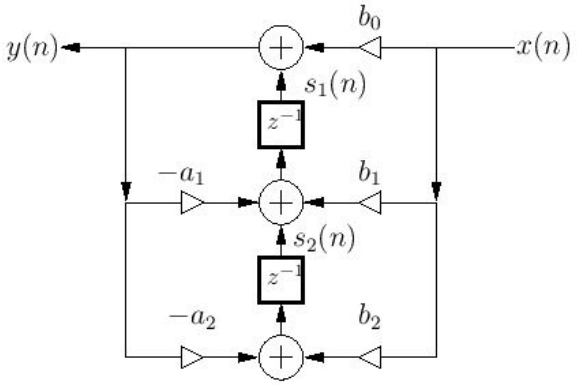


Figure 8: Filter Structure: Transposed-Direct-Form-II

1.4.4 Cascade Structure

In many situations it is better to implement a digital filter $H(z)$ in terms of first- and/or second-order elementary sections, either in series or in parallel.

In particular, such an implementation may have numerical advantages. In

the time-varying case, it is easier to control fundamental parameters of small sections, such as pole frequencies, by means of coefficient variations.

This structure reduces the effects of using finite word length representation of the filter coefficients.The cascade structure can be implemented by a series-connected of first-order and/or second-order structures, each of the 2nd order subsystems can be implemented as DFI or DFII or transposed DFII. as shown in 9.

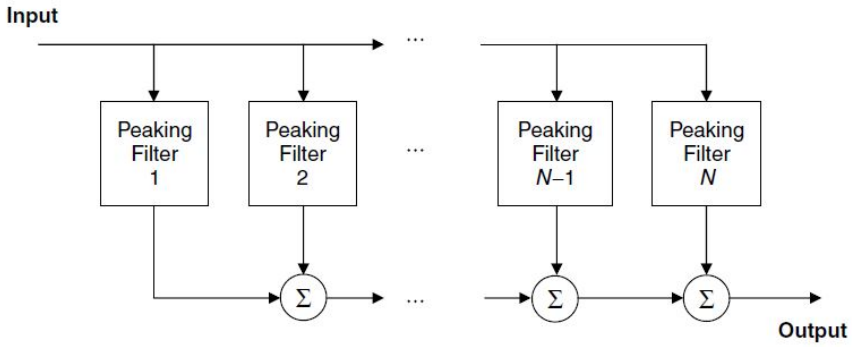


Figure 9: Filter Structure: Cascade Setup

Parallel Structure

Instead of breaking up a filter into a series of second-order sections, as discussed in the previous section, we can break the filter up into a parallel sum of first and/or second-order sections.

Parallel sections are based directly on the partial fraction expansion (PFE) of the filter transfer function.There is additionally an FIR part when the order of the transfer-function denominator does not exceed that of the numerator (i.e., when the transfer function is not strictly proper). The most general case of a PFE, valid for any finite-order transfer function.

A parallel structure of an IIR filter can be synthesized based on a partial fraction expansion performed on the transfer function H(z) as shows the

figure 10 down below

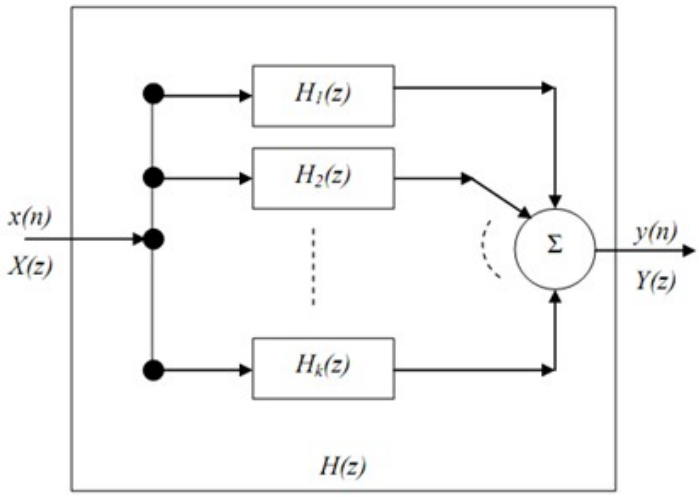


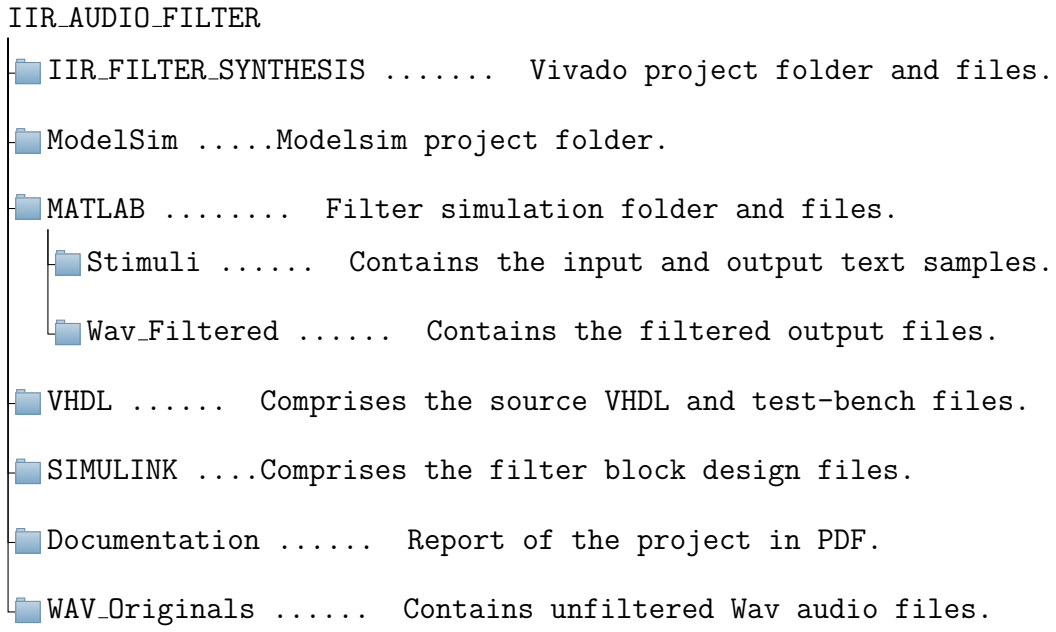
Figure 10: Filter Structure: Parallel Structure

1.5 Work Environment

In this project we used the following list of tools to achieve the requirements mentioned :

- ***Matlab***: Design , simulation and data preconditioning of the IIR filter..
- ***Simulink*** : Designing the signal flow diagram of the filter .
- ***ModelSim***: This was crucial to achieve the hardware implementation and simulate the solution.
- ***Vivado***: In order to view the synthesis logic results.

1.6 Project Structure



2 Design and Simulation

2.1 Filter Architecture and Design

Applying the shifting property of the z-transform on the system difference function (7) we can find the transfer function of the system as follows.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{-\frac{1}{4} + \frac{1}{4}z^{-4}}{1 - z^{-1}} \tag{8}$$

The **zeros** are the values when the numerator is zero and the **poles** are the values of the transfer function when the denominator is set to zero.The poles represent frequencies that cause the denominator of a transfer function

to equal zero, and they generate a reduction in the slope of the system's magnitude response.

From the transfer function the numerator coefficients are called b-vector values and the denominator values are called the a-vector values.

Hence, $b = [b0, b1, b3, ..., bM]$ and $a = [1, -a1, -a2, -a3 \ldots, -aN]$

We selected the **Transposed Direct Form II** due to the numerical numerical robustness of it previously mentioned, especially in fixed-point implementation, in fact it is canonical with respect to delay compared to DFI. This happens because delay elements associated with the two-pole and two-zero sections are shared, as well as avoid the avoiding the internal overflow of DF-II that could happen.

The signal flow diagram of the IIR filter that was implemented is conveyed below in the figure 11

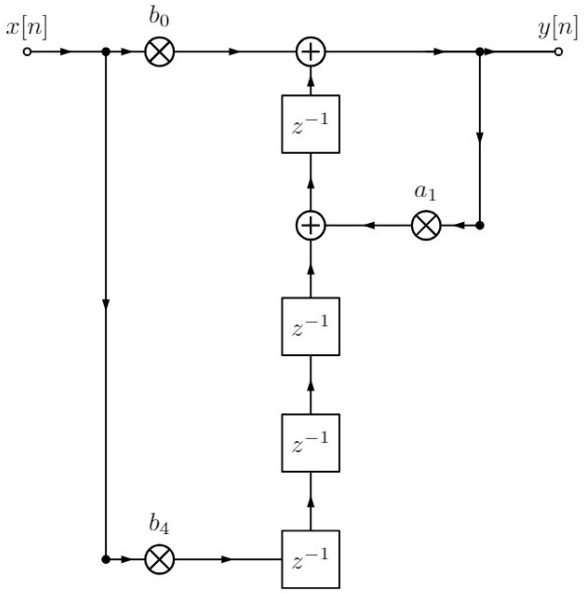


Figure 11: TDF-II Signal Flow Diagram

2.2 Filter Simulation

Through the medium of Matlab we were able to simulate the exact same filter design on it and consult the multiple responses of the filter to our stimuli. We implemented the following IIR filter using fixed point arithmetic that depicts the same requirements stated in task.

Through the filter frequency response and phase response in Matlab it turns out that we are dealing with a *Notch Filter* , as depicted in the figure 12 and 13 below :

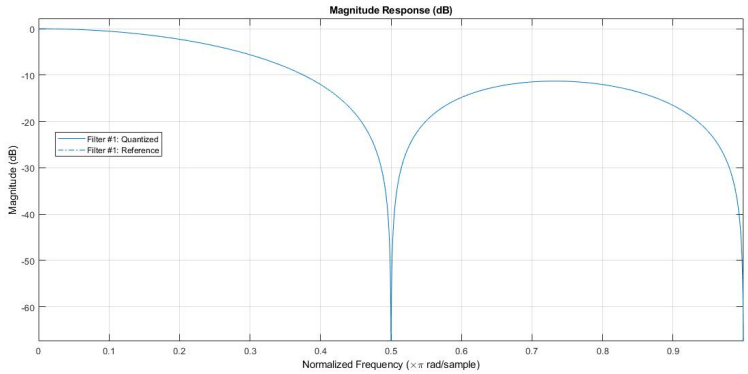


Figure 12: IIR Filter Frequency Response Using Matlab

The stability of an IIR filter can easily be inferred from the zero-pole plot of the filter's transfer function. As a result a digital filter is stable if and only if the poles of the irreducible filter function lie inside the unit circle in the z-plane.
Any pole outside of this unit circle introduces the instability of the filter. In this case the filter impulse response increases exponentially. If the transfer function has poles on the unit circle then the impulse response neither decays nor grows as 'n' goes to infinity.

The Impulse response $\delta(n)$ of the filter and its z-plane plot of poles and zeros are shown below, it is important to note here that actually the filter is not

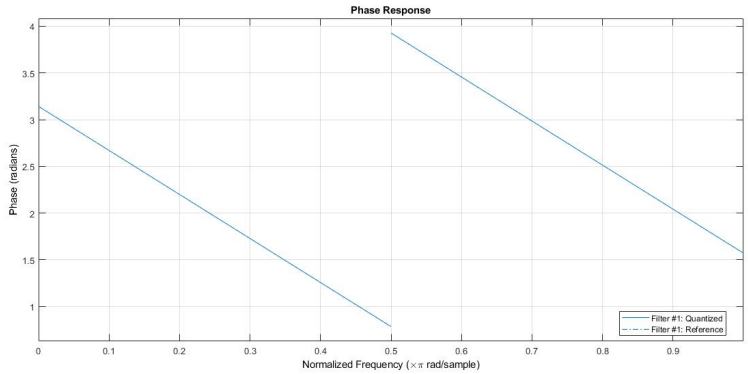


Figure 13: IIR Filter Phase Response Using Matlab

stable for large calculations as the poles are supposed to be strictly inside the unit-circle which is not the case as depicted in the figure 14.

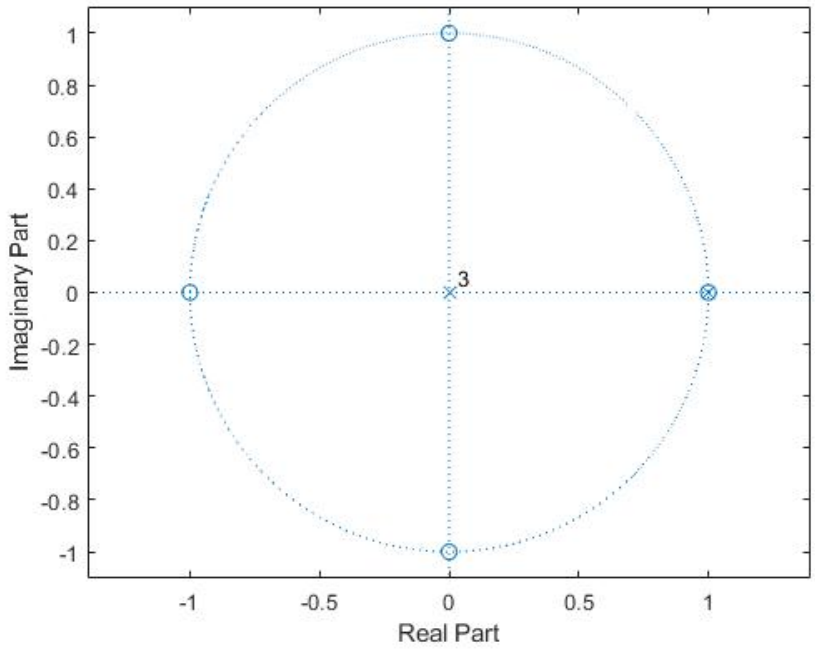


Figure 14: Poles and Zeros in the Z-Plane

2.3 Stimulus Data Generation

Alongside the simulation of the actual IIR filter on Matlab we are going to act with that to prepare and put into place a reference filtering stimulus that will be the base of the test-plan and simulation of the digital circuit later in its hardware implementation.

Since we were advised to implement an application to a real-world data on top of 16 bit mono WAV file format, we used the Matlab in order to derive from the WAV files multiple samples and to apply the filtering method on them in order to acquire the end result, in fact we used three main WAV files : Siren.wav, Thunder.wav and Trafficjam.wav and their results are accompanied within the project files for consulting (we chose that the sampling frequency is to be preserved) .

Two text files for each WAV audio file that withhold at first the input data demonstrated through 16 bits fixed point arithmetic samples and their respective filtered output that will be used as the correct reference, this stimulus data generated will be used in the test-bench on VHDL simulation phase, a snapshot of the Matlab code is displayed below :

```
1  %% This Matlab script serves the purpose of simulating an IIR FILTER of ...
       the following difference equation : y[n] = y[n - 1] - 1/4 x[n] + 1/4 ...
       x[n - 4]
2
3  %% ---------------------------- FILTER SIMULATION ...
       ---------------------------- %%
4
5  num_coeff = [-1/4 0 0 0 1/4];
6  den_coeff = [1 -1 0 0 0];
7
8  iir = dfilt.df2;
9  set(iir, 'arithmetic', 'fixed', ...
10     'OutputMode', 'SpecifyPrecision', ...
11     'Numerator', num_coeff, ...
12     'Denominator', den_coeff, ...
13   ......
14
15     'CastBeforeSum', false);
16
17 %% This command launches the Matlab fvtool which allows to monitor the ...
       magnitude and phase response of the
18 fvtool(iir);
```

Listing 1: IIR Filter Simulation on Matlab

3 Hardware Implementation and Testing

3.1 VHDL Fulfilment

As shown previously, the signal flow diagram was coded in VHDL and simulated in Modelsim. The final resolution was the following : 4 registers that were used as the elementary delayers and a ripple adder carry in addition to a ripple borrow substractor.

We are ought to note here that the multipliers were not accomplished as an external components, this is intended since in this special case it is depicted already that the coefficient multiplication could have easily done through a shifting process, reminding here that a N arithmetic right shift is a division by 2^N. In fact *1/4 * sample(0)* is a two bit arithmetic right shift.

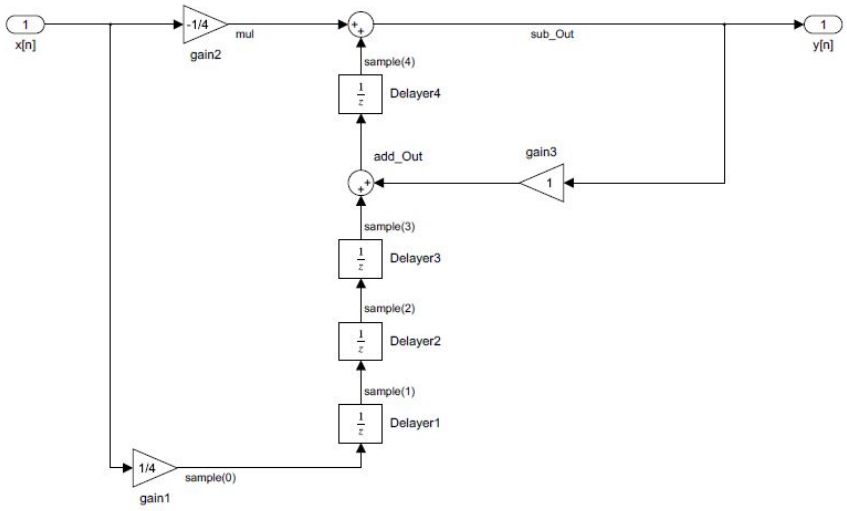


Figure 15: Schematic diagram of the IIR filter Implementation of 7 in SIMULINK

We did not account here for the flexibility of the filter since the task was deliberately and clearly stated, and we worked on the specific difference equation

provided.The final hardware implementation diagram is depicted in fig. 15

We notice here that the internal connections between the elements are on 18 bits and this is in order to account for the overflow inside the circuit which already the TDF-II handles well, the final output of the filter is resized again to 16 bits by truncating the least significant bits.

3.2 Filter Testbench

Generally speaking, a VHDL test-bench is an important part of VHDL design to check the functionality of the design through simulation waveform.

In our case we used these tesbench files to check if the filter designed in MATLAB conforms with the implementation in the VHDL part.

In this project there are **4** test bench files. As stated in the design part we used as input to the filter the following four signals: the **Impulse Response**,the **Siren** input, the **Thunder** and the **Trafficjam** files.

The test took place by reading from locally stored files as inputs and expected outputs as arrays of samples using the **std.textio library** and feeding the filter with a new sample on each clock period, if an inconsistency occurs the sample line number will be reported.

Impulse Response

While doing the testing of our design the first task was to check if the impulse response of the filter, which is one of the crucial parameters, is as expected.

To that effect a 5-element array is chosen and given as an input to the filter in MATLAB.Hence, the impulse_in.txt and impulse_out.txt files.Then these files were used in the VHDL testbench code for verification using the **assertion** function.

If no error messages was shown, then the designed filter in MATLAB and the implemented one in MODELSIM have the same behavior and we are good to go.The simulation output for the impulse response of the filter is shown in fig 16.

$$\delta = [\ 1 \quad 0 \quad 0 \quad 0\ ]$$

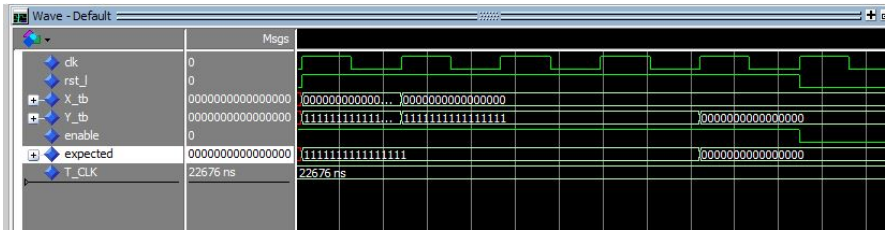


Figure 16: Impulse response test bench

```
1  entity IIR_Impulse_tb is
2  end IIR_Impulse_tb;
3
4  architecture arch of IIR_Impulse_tb is
5  ...
6      begin
7      file_open(file_INPUT, "C:\myvhdl\iir_inf\golden_ref\impulse_in.txt",  ...
           read_mode);
8      file_open(file_OUTPUT, ...
           "C:\myvhdl\iir_inf\golden_ref\impulse_out.txt", read_mode);
9          rst_l <= '0';
10         wait until clk'event and clk='1';
11         rst_l <= '1';
12         while not endfile(file_INPUT) and not endfile(file_OUTPUT) loop
13             i:=i+1;
14             readline(file_INPUT, v_ILINE);
15             read(v_ILINE, v_ADD_TERM1);
16             readline(file_OUTPUT, v_OLINE);
17             read(v_OLINE, v_ADD_TERM2);
18             X_tb <= v_ADD_TERM1;
19             expected <= v_ADD_TERM2;
20             wait until clk'event and clk='1';
21             assert (Y_tb = expected)
22             report "Inconsistency at line number : " & integer'image(i)
23             severity error;
24         end loop;
25         end process;
```

Listing 2: Impulse response testbench using assertion in VHDL

WAV Data Samples

To check the correctness of the filter we handled different test WAV files present on a large frequency bandwidth.As we already stated, we extracted an input data and a reference output ones from the WAV files using Matlab in order to use them as a reference for the testing, which ran successfully without any errors depicted. The samples are fed into the VHDL simulation via text files containing in each line a 16-bit fixed point arithmetic value.

The files used are four in number: the **siren**,**thunder** and **trafficam**. The simulation outputs of the aforementioned files are depicted in fig. 17, fig.18 and 19 respectively.

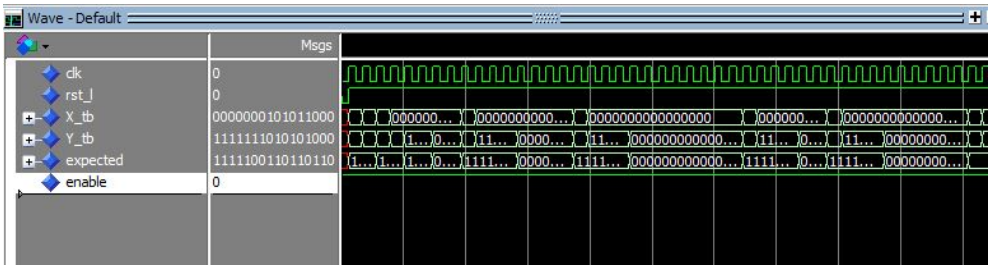


Figure 17: Siren WAV Testbench Results

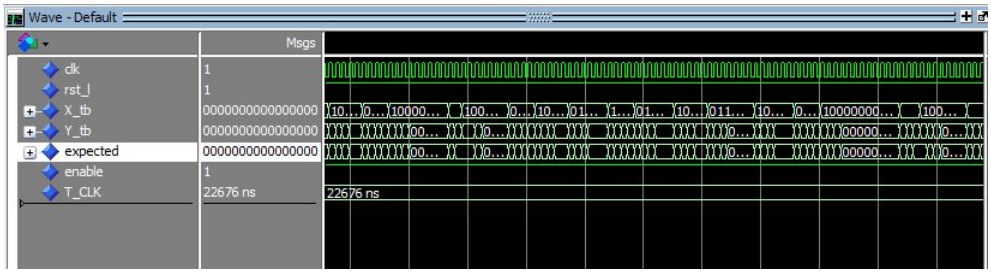


Figure 18: Thunder WAV Testbench Results

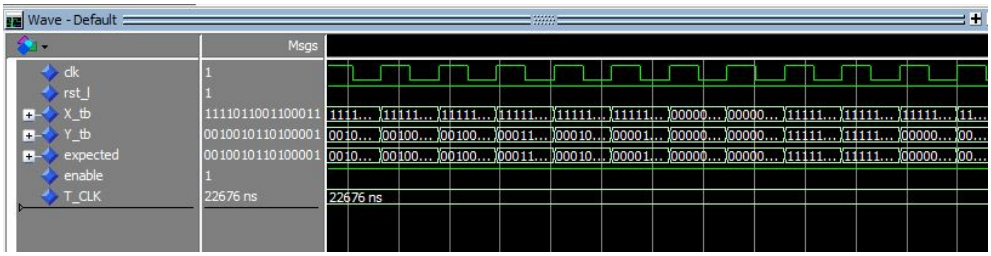


Figure 19: TrafficJam WAV Testbench Results

4 Vivado Synthesis Logic Results

4.1 RTL Design

As a final step in this project development we had to synthesize the VHDL hardware implementation using Vivado on top of an FPGA Xilinx Zybo board model number xc7z010clg400-1 following the steps in [5].

The RTL Analysis reported the elaborated design schematic shown in 20 :

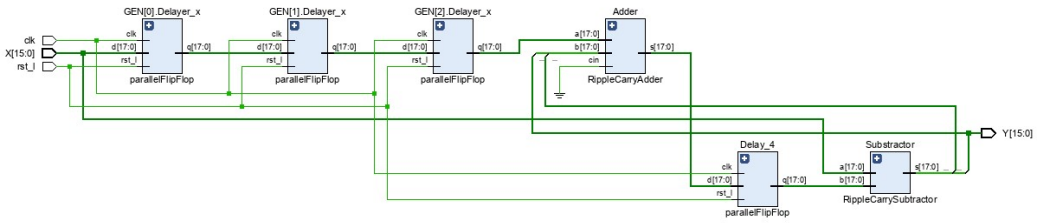


Figure 20: RTL Final Design Schematic

According to the RTL Design 6 cells 34 I/O Ports and 127 Nets were applied.

The DRC in has reported 3 violations, a warning and 2 critical warnings.

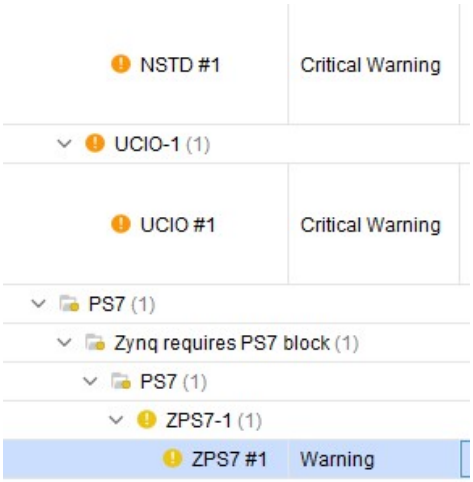


Figure 21: DRC Report Violations

Warnings **NSTD-1** and **UCIO-1** are according to Xilinx to notify users that they need to set IO-STANDARD and PACKAGE-PIN, in order to protect devices from accidental damage that could be caused by the tools randomly choosing a pin location or IOSTANDARD without the knowledge of the board voltage or connections. In our case it's due to the fact that we haven't set the Pin placement on the board in the constraints file.

The next violation says that The PS7 cell must be used in this Zynq_7000 design in order to enable correct default configuration, which refers to the ARM Processor environment that we haven't used in this process and it can be ignored.

4.2 Synthesis Results and Timing Constraints

In digital audio, 44,100 Hz is a common sampling frequency, analog audio is often recorded by sampling it 44,100 times per second. In order to obtain the timing report we had to specify a clock period constraint of $T_{ck} = 22676ns$.

The synthesis design ran successfully with no violations whatsoever observed, and all of the user made constraints were met as the *Worst Negative Slack* is of 22669,664ns which is also the **Critical Path** which relies Delay_4/q_reg[2]/C to Delay_4/q_reg[14]/D which is the present in the *Delayer 4* ParallelFlipFlop as shown in the figure 22 !

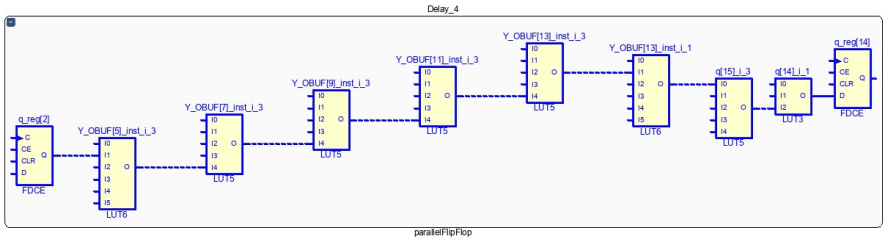


Figure 22: Critical Path Showcase

The *Worst Hold Slack* was at 11337,499ns. The complete time report results are shown in fig. 23 :

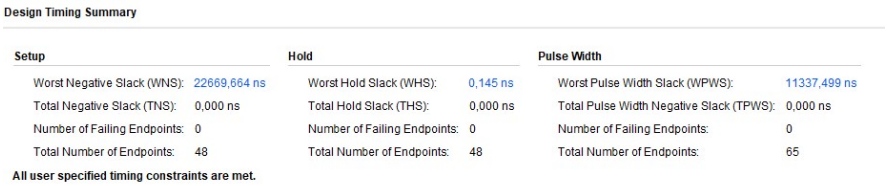


Figure 23: Time Report Summary

The maximum frequency a design can run on Hardware in a given implementation is 9

$$F_{max} = \frac{1}{T_{ck} - WNS} \tag{9}$$

With WNS positive or negative and T as the target clock period.

According to the equation 9 the maximum frequency the we could run on this digital circuit is the following

$$F_{max} = \frac{1}{T_{ck} - WNS} = \frac{1}{(22676 - 22669,664) \cdot 10^{-9}} = 157.82MHz \tag{10}$$

The Place and Route step hasn't neither reported any warning or violations, the figure 24 depicts the implemented circuitry on FPGA fabric.

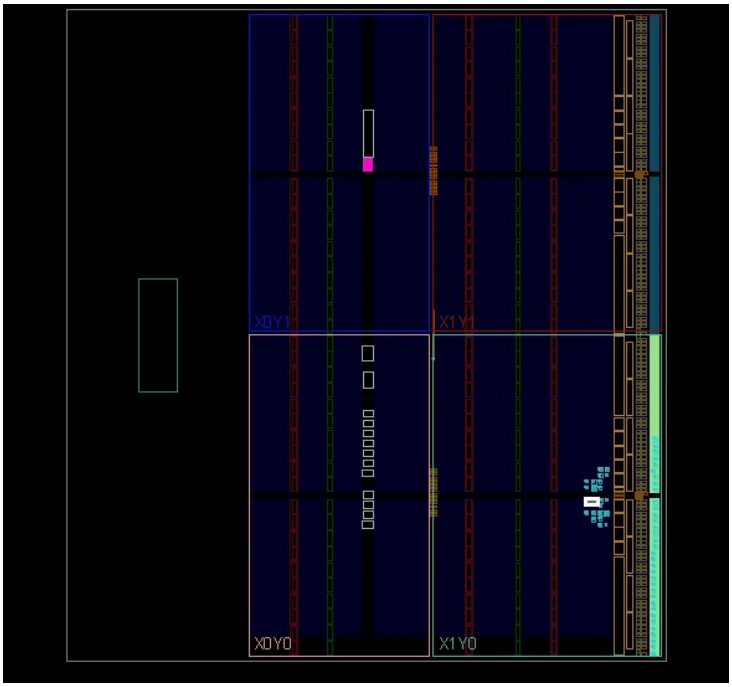


Figure 24: Implementation on FPGA Fabric

4.3 Utilization and Power Reporting

The project that we have carried on the Zybo board was not heavy on the board and in addition to the optimizations that we followed mainly while using the TDF_II the utilization of the FPGA fabric was little to nothing, in fact less than 1% of the LUT tables and FlipFlips were used as well as 34% of the I/O, figure 25 depicts clearly the usage.

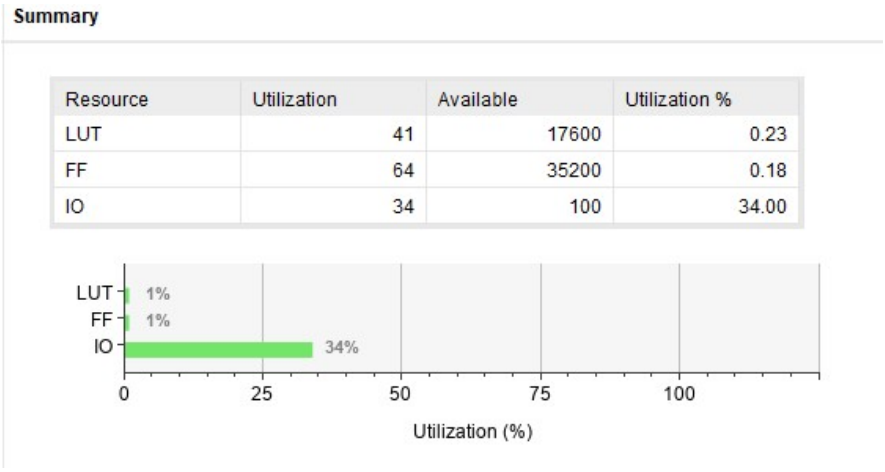


Figure 25: Resources Utilization Report

Although it not reliable, Vivado tools tries to approximate the power usage by the board during the process of filtering the figure 26 showcases the power utilization report.

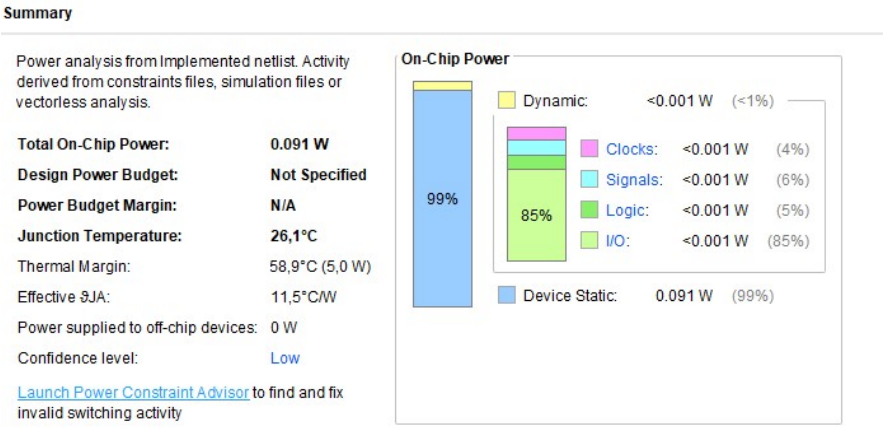


Figure 26: Vivado Power Report Summary

5 Conclusion and Perspectives

In conclusion, the project we carried out gave us a brief introduction to the field of digital filtering and to explore its different applications in several domains.

We had the chance to work the digital circuit out throughout simulating and implementing to testing, despite the fact that filters are fairly easily understood and calculated, the practical challenges of their design and implementation are significant and are the subject of much advanced research.

As a perspective to follow up with our project we would mention generating the *bit stream* and applying the designed filter on real-world audio application since the task was not required in this project.

References

[1] I. Jervis, "Digital signal processing: A practical approach (2nd edition)," 2001.

[2] U. Meyer-Baese and U. Meyer-Baese, *Digital signal processing with field programmable gate arrays*, vol. 65. Springer, 2007.

[3] V. A. Andreas Spanias, Ted Painter, *AUDIO SIGNAL PROCESSING AND CODING*. Wiley, 2007.

[4] B. Ryan, "Wav audio format wikipedia," 2022.

[5] P. Luca Fanucci and D. Massimilano, *Electronic Systems Lecture Slides*.