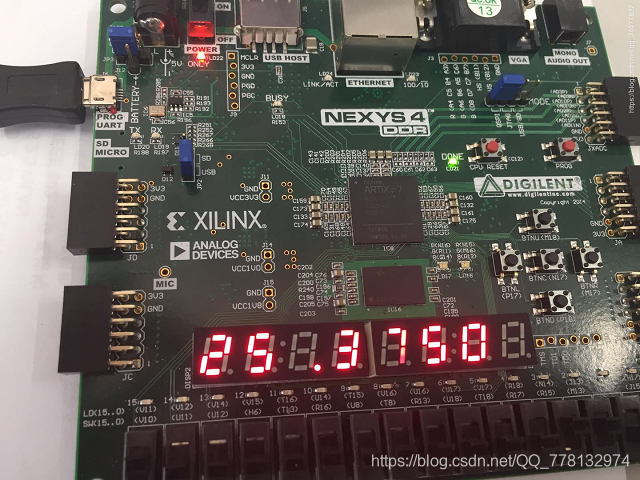


UE: Digital electronics

# **Digital Electronics Project**



### KADDOURI Nassira

### AITELHADJ Yassmine

### *Département:* EEEA

*Major:* Electronics

*Year:* 2024

*Groupe:* J

**Introduction**

The objective of this project is to design a complete instrumentation chain, covering the acquisition and digital signal processing. The project aims to measure the frequency of a real-time input signal using the Artix-7 FPGA from Xilinx.

The requirements specify that the input signal will have a frequency range between 10 kHz and 100 kHz. A high-pass filter with a cutoff frequency of 10 kHz must be implemented to eliminate low-frequency components. Additionally, the measured frequency should be displayed on the 7-segment display provided with the Artix-7 board.

The proposed architecture for the project is illustrated in Figure 1.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 1 : Organizational chart of the project

1. **Data acquisition and validation of the ADC**

In this part, we validated the functionality of the integrated ADC in the Artix-7, where we implemented our frequency meter project. To achieve this, we used a DAC based on an R2R ladder structure connected to one of the PMOD connectors. This type of DAC, being passive, was simple to implement and required feeding it with digital data refreshed at the appropriate rate, matching the ADC's sampling rate. We first tested the DAC by feeding it a ramp signal, which allowed us to successfully display a sawtooth signal of 255us period at its output, confirming its proper operation.

* 1. **Analogic to digital conversion**

The goal of the first part is to convert an analog signal into a digital signal. This allows us to process the signal and retain only the characteristics necessary for frequency analysis. The initial approach involves using the FPGA's integrated converter, called the XADC. Using the Xilinx Analog-to-Digital Converter (XADC) Wizard in LogiCORE IP, an HDL file was generated to configure the converter.

Une image contenant texte, logiciel, nombre, Police

Description générée automatiquement

Figure 2: configuration of the XADC

According to Figure 2, the DRP interface (default) is initially considered. The Startup Channel Selection is configured for single-channel mode, allowing us to monitor only one channel. The Timing Mode is set to Event Mode, meaning an external trigger is required to start the conversion. The ADCCLK is derived from the DRP clock and must fall within the range of 4–26 MHz. The Vivado IDE calculates the clock divider based on the desired conversion rate, which is set to 1 Msps, and the DCLK frequency of 100 MHz.

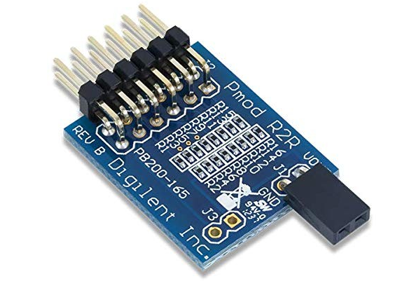
The transfer function can be obtained from the XADC datasheet. It shows that a 12-bit ADC converter is used with an amplitude of 1V. Code transitions occur in successive values of the LSB (Least Significant Bit), where LSB = 1V/1012 = 244 µV.

Une image contenant Composant de circuit, Composant électronique, Composant de circuit passif, Ingénierie électronique

Description générée automatiquement

We aim to use the input signal in differential mode, which requires always having a positive input (Vp) and a negative input (Vn). We will use the Digilent PmodTPH2 module, which allows us to apply the differential input to the ADC.

**1.2. Digital to analogic conversion**



To validate the previous converter, we focus on the inverse conversion. We will use an R-2R DAC.

Unfortunately, the Pmod R2R only operates with 8 bits. Therefore, we will use only the 8 most significant bits (MSB) from the ADC output register, as the MSBs contain the shape of the signal, while the least significant bits (LSBs) are mainly for precision.

To properly verify the output, the DAC needs to be fed, and the digital data must be converted. This conversion should ideally occur at the same sampling rate as the ADC. Therefore, we will create a timing block (tempo) to synchronize them.

After combining them, we generate a 10 khz signal that has positive and negative sinusoids, give it as input of the ADC through the PmodeTPH2 and visualize the output of the DAC in the oscilloscope. And Indeed, we fine that the input of our system and the output are identical as shown in the following figure, thus the validation of the proper functionality of the ADC.

Une image contenant texte, capture d’écran, Logiciel multimédia, Tracé

Description générée automatiquement

Figure 3: validation of the ADC with reverse conversion

1. **Signal conditioning**

**2.1 Filter**

We want to design a first-order high-pass filter with a cutoff frequency of 10 kHz.

The transfer function of such filter:

Where: τ is the time constant of the filter, related to the cutoff frequency ​ .

we start by relating the output Y(p) and input X(p) using the transfer function:

Next, we multiply both sides by to eliminate the denominator:

Taking the inverse Laplace transform of both sides, where p corresponds to we get the

differential equation:

When we discretize time, we set as the input at the instant and as the output at the instant , where and is the time step.

We then obtain

Rewriting this equation gives:

We can replace by , if and

Then

We were able to implement the recursive equation of the filter in VHDL by creating two blocks: one for **memory** and one for **calculation**:

**Memory Block**: This block stores the previous values of the input and output signals and , which are needed for the recursive calculation.

**Calculation Block**: This block performs the filter's recursive computation using the previous equation to calculate the current value of the output.

We decided to encode each number using 9 signed integer bits and 13 fractional bits, for a total of 22 bits (21 down to 0). The 9 signed integer bits correspond to the 8 bits of the input signal plus 1 sign bit. The 13 fractional bits allow us to ensure a precision up to the ten-thousandth place (with a conversion error smaller than For example, would be encoded as "0010 1101, 1010 1011 0000 0".

We Displayed the output of the filter in the oscilloscope and measured its cutoff frequency with the method of 7/5 caree, we found khz, close to 10 khz. This difference might be induced by the limited precision of the calculation or the discretization of data.

Une image contenant texte, capture d’écran, Logiciel multimédia, logiciel

Description générée automatiquement

Figure 4: response of the filter to a square shaped signal

Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement

Figure 5: response of the filter to a sinusoidal Input

**2.2 Differentiator (Derivateur)**

We are now tasked with implementing a differentiator. The system input is more like a triangular signal, and the differentiator will help transform the slopes of the triangle into constants, which will make it easier to detect the beginning of a period.

To implement the differentiator, we need to perform the following operation:

By discretizing this, we obtain:

To avoid a complicated multiplication by that only influences the amplitude of the signal, while keeping the filter's output significant, we propose replacing the factor with 4, which gives the following equation:

Thus, the differentiator principle becomes a simplified version of the filter.

To test this differentiator, we applied a triangular signal with a 100% duty cycle (sawtooth signal). The output waveform, as observed on the oscilloscope, is shown in Figure 6.

Une image contenant capture d’écran, texte, Logiciel multimédia, Tracé

Description générée automatiquement

Figure 6: response of the derivative to a sawtooth signal

The output is a constant that corresponds to the slope of the sawtooth signal given in the input, which proves that the differentiator effectively outputs the derivative of its input.

**2.3. Thresholding (Esseuillage)**

This part is very simple. We just need to remove the offset that we added in order to adapt our output signal’s representation to the DAC’s encoding, then we should compare the current value of our signal to . is our threshold.

This Bloc will provide as with a sort of pulses whenever the input signal goes beyond This output is going to help us calculate the period of the signal easily.

**Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement**

Figure 7: Output of the thresholding