CentraleSupélec

FPAA Project

Lab Report
NMOS Neuron on the 3.0a FPAA

Authors:
Corentin Tafani
Joseph Brisson

7th June 2019

CentraleSupélec

Acknowledgments

We would like to thank all the members of this project on the FPAA board. Thank you Yassine, Bertrand, Hugo and Benjamin for this wonderful year. All these days at GTL were not vain and we can be proud of what we have produced.

We also would like to thank our project advisors, M. Locquet professor at Georgia Tech Lorraine and M. Rontani professor at CentraleSupélec. They advised us during the project and guided our work.

We would like to thank Mrs. Hasler professor at Georgia Institute of Technology for sending us the board, without which there would have not been any project.

Finally, we would like to thank Mrs. Aishwarya Natarajan Phd student at Georgia Institute of Technology for helping us with the board.

Contents

List of Figures

Introduction

This report presents our work on the FPAA 3.0a Board. We implemented and tested a Neuron Design. The idea is to use the non-linearity in voltage of a CMOS Transistor. Our final objective is to have a working reservoir to execute machine learning tasks on the FPAA. This report will also introduce FG and CMOS transistors. We studied them prior to the study of the neuron. We used CMOS Transistors as the main elements of our node and our first idea of design was to use a FG transistor to have a low-pass filter. This report will show how and why the design of our node evolved.

1 Presentation of our work

1.1 Framework of the project

The FPAA board allows the programming of complex low-voltage analog circuits. The main components are embedded in CAB (Analog Blocks). The cards relies on a Floating Gate Transistor Routing Net. To obtain details about the structure and the programming of the FPAA, I refer you to our main report and guide of use of the FPAA.

1.2 Our approach to Reservoir Computing on a FPAA

A first part of our work consisted of several components characterization. We were searching for a components with an interesting non-linearity. This report contains a brief presentation of this work to get an idea of the way the board works. We finally choose a CMOS transistor based design. We implemented the non-linearity and characterized it. The next step was to find a way to add a delay and finally testing the node on a decision feedback equalization (DFE) problem.

2 An overview of some analog components available on the FPAA

2.1 CMOS Transistor

CMOS transistors are available on the board. We use this transistor to have a non-linear relation between the input and output voltage. Our main objective was to see if we could use the sub-threshold behavior of the transistor.

As a reminder, here is the characteristic of a MOSFET transistor on figure 1. The EKV model is a more precise description of the component characteristic (figure 2). The Modelicka block library is available on the panel of Scilab (figure 3) and you can visualize how relevant is the EKV model to characterize at subthreshold voltage or above threshold voltage. Figure 4 presents the results of the simulation for above-threshold voltage and figure 5 presents the results for sub-threshold values.

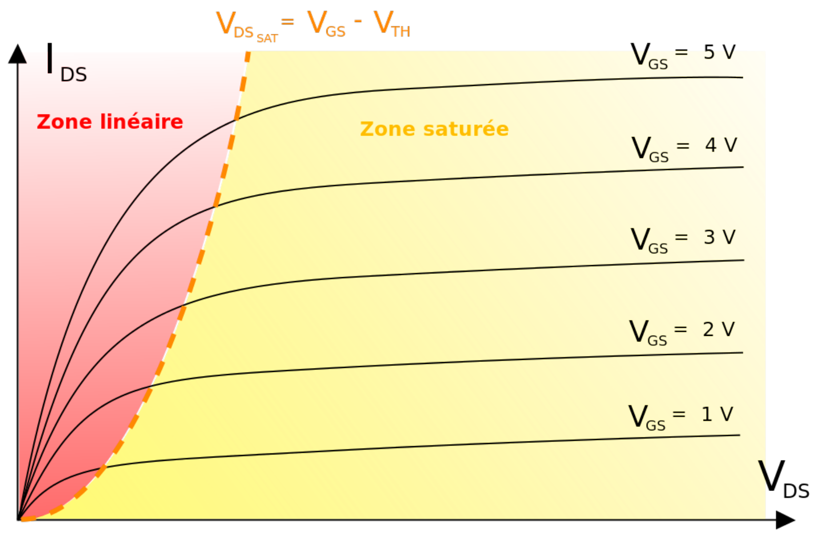

Figure 1: Characteristic of a MOSFET

Sub-threshold values has an interesting non-linear behavior but we wanted something that is more like a sigmoïd. We finally choose a design based on two N-MOS transistors. One of the transistor source is connected with

	nFET	pFET
EKV	$I_{th}\ln^2(1+e^{(\kappa(V_g-V_{T0})-V_s+\sigma V_d)/2U_T})$ $-\ln^2(1+e^{(\kappa(V_g-V_{T0})-V_d+\sigma V_s)/2U_T})$	$I_{th}\ln^2(1+e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))/2U_T})$ $-\ln^2(1+e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_d)+\sigma(V_b-V_s))/2U_T})$
For sub V_{T0}: Ohmic Saturation	$I_{th}e^{\kappa(V_g-V_{T0})/U_T}(e^{-V_s/U_T}-e^{-V_d/U_T})$ $I_{th}e^{(\kappa(V_g-V_{T0})-V_s+\sigma V_d)/U_T}$	$I_{th}e^{\kappa(V_b-V_g-V_{T0})/U_T}(e^{-(V_b-V_s)/U_T}-e^{-(V_b-V_d)/U_T})$ $I_{th}e^{(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))/U_T}$
For Above V_{T0}: Ohmic Saturation	$\frac{I_{th}}{4U_T^2}((\kappa(V_g-V_{T0})-V_s)^2-(\kappa(V_g-V_{T0})-V_d)^2)$ $\frac{I_{th}}{4U_T^2}(\kappa(V_g-V_{T0})-V_s+\sigma V_d)^2$	$\frac{I_{th}}{4U_T^2}((\kappa(V_b-V_g-V_{T0})-(V_b-V_s))^2-(\kappa(V_b-V_g-V_{T0})-(V_b-V_d))^2)$ $\frac{I_{th}}{4U_T^2}(\kappa(V_b-V_g-V_{T0})-(V_b-V_s)+\sigma(V_b-V_d))^2$

Figure 2: EKV Model

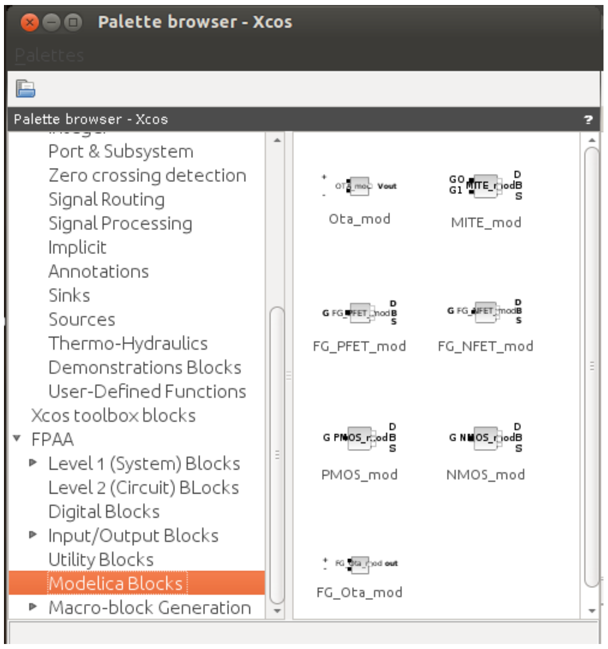

Figure 3: Modelicka in Scilab

the other transistor drain. The details of this design is given in the part describing our final design choice.

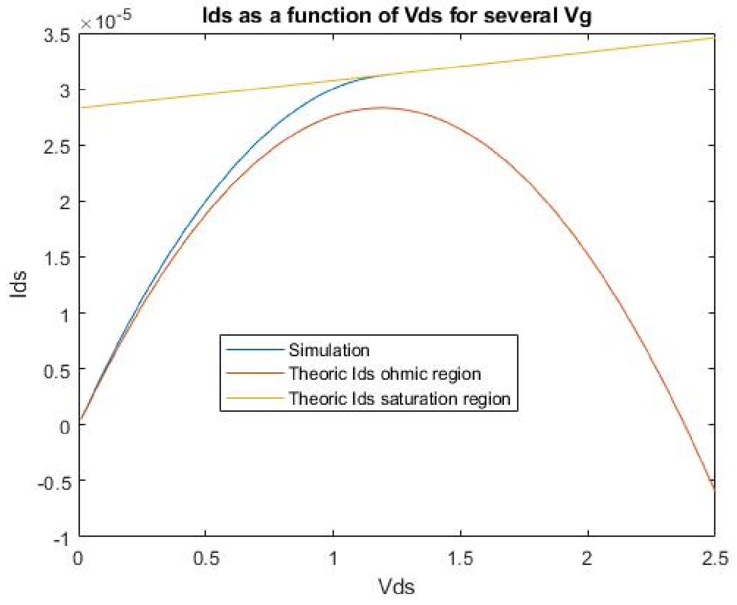

Figure 4: Above Threshold Characteristic of the CMOS with the Modelicka Block with I_{DS} as a function of V_{DS}

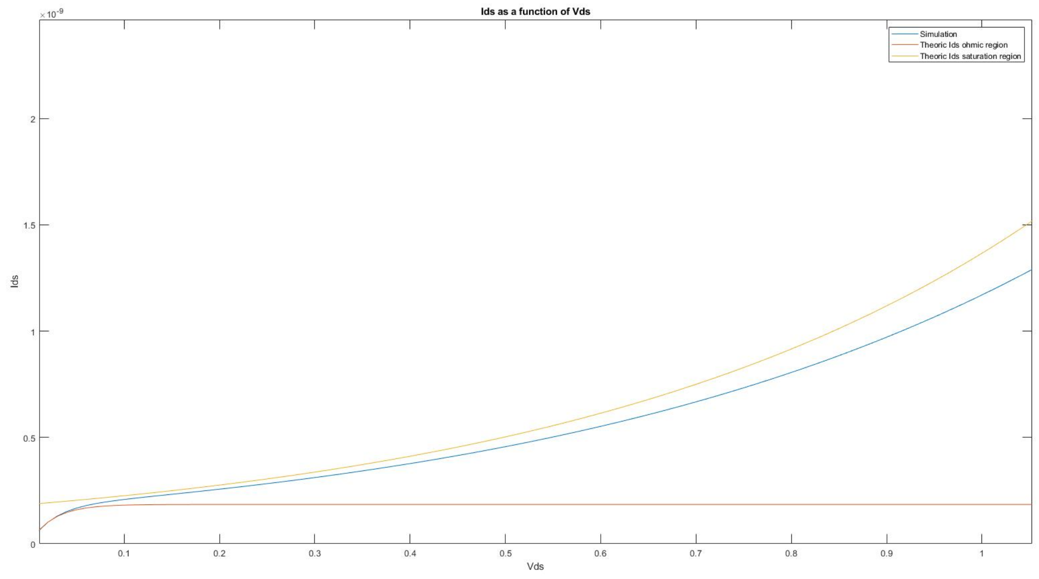

Figure 5: Sub Threshold Characteristic of the CMOS with the Modelicka Block with I_{DS} as a function of V_{DS}

2.2 Floating Gate Transistor

2.2.1 Presentation of the FG Transistor

Floating Gate Transistors and MITE are available on the FPAA board.

A floating-gate transistor is based on a classical MosFet architecture. The base is modfied by adding a floating gate between two layers of isolation, the interesting characteristic of this layer is that it keeps its net charge even when no current is entering the transistor. On top of it, there are control gates layers that are inputs.

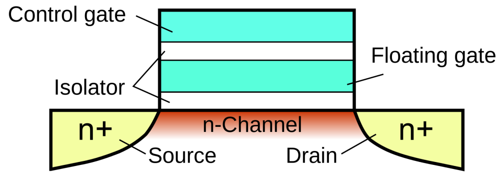

Figure 6: Floating-Gate Transistor

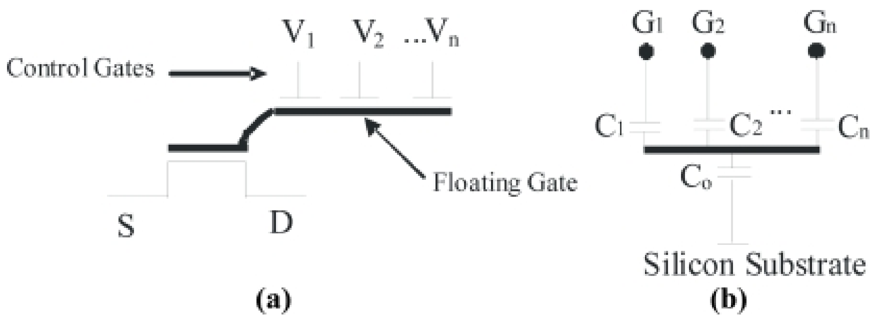

Figure 7: Layers

Then, the DC equation modeling the operations of the transistor are the same than the MOSFET. The only difference being the voltage on the gate depends on the control gates and the charge in the floating gate. The charge of the floating-gate Q_{FG} is fixed by particular processes (increasing with tunneling or decreasing with hot-carrier injection).

$$V_{FG} = \sum_k V_k \frac{C_k}{C_T} + \frac{Q_{FG}}{Q_T}$$

avec V_k aux inputs de contrôle.

We can see on this graph that depending on the charge in the floating gate, it changes the V_t the voltage of threshold and the relation between the saturation current and the voltage at the gate. On figure 9, you can see the results with the MITE Modelica Simulation block.

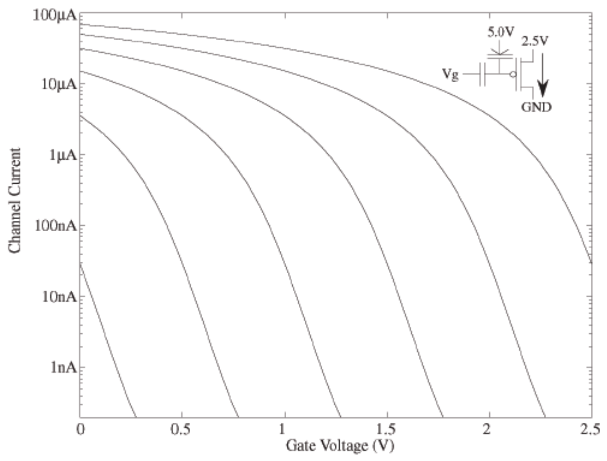

Figure 8: Saturation current/Gate voltage relation with different charge of the floating-gate

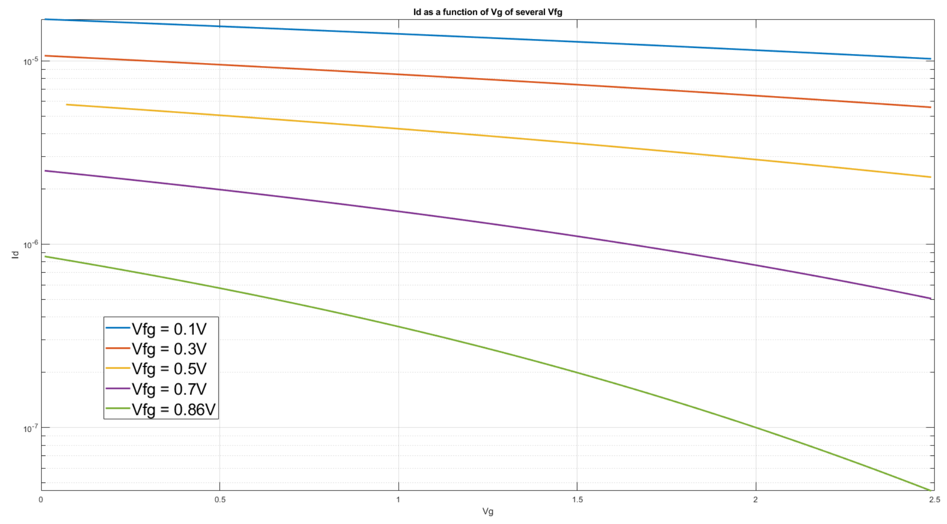

Figure 9: Results using the simulation MITE block

2.3 Making a resistor with FG transistor

To add a delay between the input and the output of our node, we wanted to implement a low-pass filter. Our first idea was to use one of the capacitor of the board with a floating-gate transistor used as a resistor. This design is developped in A Functional MOS Transistor Featuring Gate-Level Weighted Sum and Threshold Operations (Tadashi Shibata, Member, IEEE, and Tadahiro Ohmi, Member, IEEE).

10.

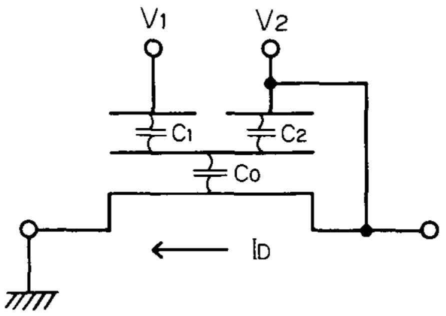

Figure 10: Circuit of the FG Transistor resistor

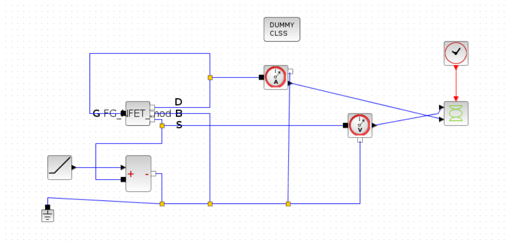

Figure 11: Scheme of a resistor made using a FG-NFET

The parameter V_{FG} has an influence on the results. The greater V_{FG} is, on a wider range of value the resistance works. For example, the figures 12 and 13 show the results for $V_{FG} = 1V$ and $V_{FG} = 2V$. As we can see, the output is linear on the range $[0, V_{FG}$ approximately. The equivalent resistor obtained is $2,9.10^{13}\Omega$ for $V_{FG} = 1V$ and $1,1.10^{13}\Omega$ for $V_{FG} = 2V$. So V_{FG} does not have a lot of influence on the value of the equivalent resistor we can make. So, the issue with this design is that we could not really decide the value of our resistor. As w can see on figure 13, the resistance is around $10^{13}\Omega$. Having only seven types of capacitors available on the card with values around 100pF, it will not allow us to make efficient low-pass filters as the highest cut-off frequency in theory would be around 0.1Hz. We decided to choose a design based on OTA's to have our low-pass filter.

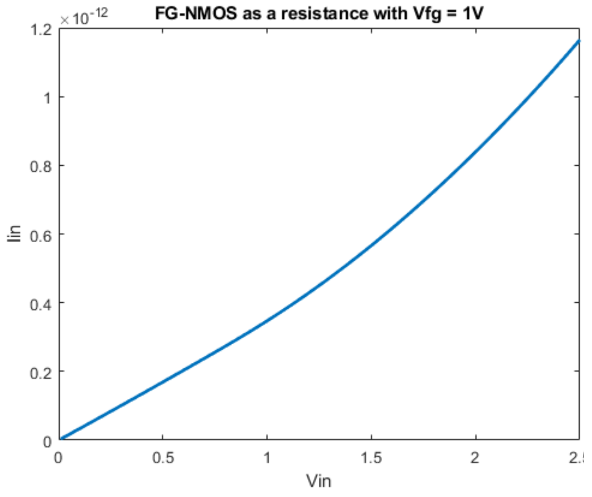

Figure 12: $V_{FG} = 1V$

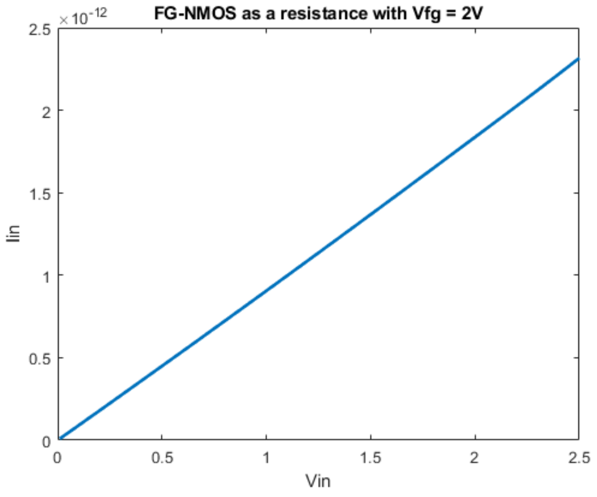

Figure 13: $V_{FG} = 2V$

3 A neuron based on NMOS

3.1 The core of the neuron : Using the NMOS non-linearity

We tried to implement a design of neuron made of 2 NMOS transistors and proposed by Son-Dorian NGUYEN. The design is shown on figure 14.

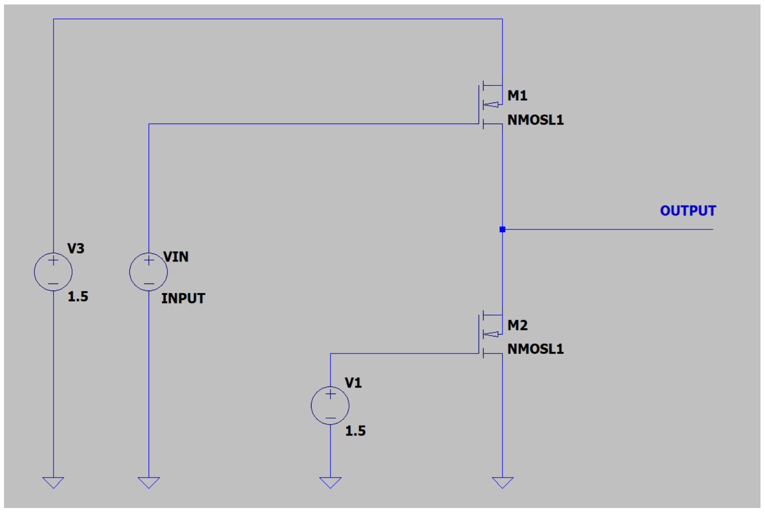

Figure 14: Scheme of the neuron

First, we simulated this neuron with the modelica blocks of the board. The aim is then to use this neuron design in on-chip or off-chip experiments. Since we did not manage to program the neuron design on-chip and it was successful off-chip, we decided to use $V_1 = V_3 = V_r$. Indeed, this choice is convenient since we only have 2 waveform generators on the digilent tool. So we decided to use one for the input and the other for V_r.

On figure 15, we tested the neuron in simulation with the following parameters:

- the input is a ramp covering the whole usable voltage value range, i.e $[0, 2.5]V$

- we tested for several V_r: 0.5V, 0.7V, 1V, 1.2V, 1.5V

As we can see, V_r has an influence on the look of the non-linearity. First, it changes the amplitude of the non-linearity. Moreover, when V_r is higher, we can see that the non-linearity saturates later.

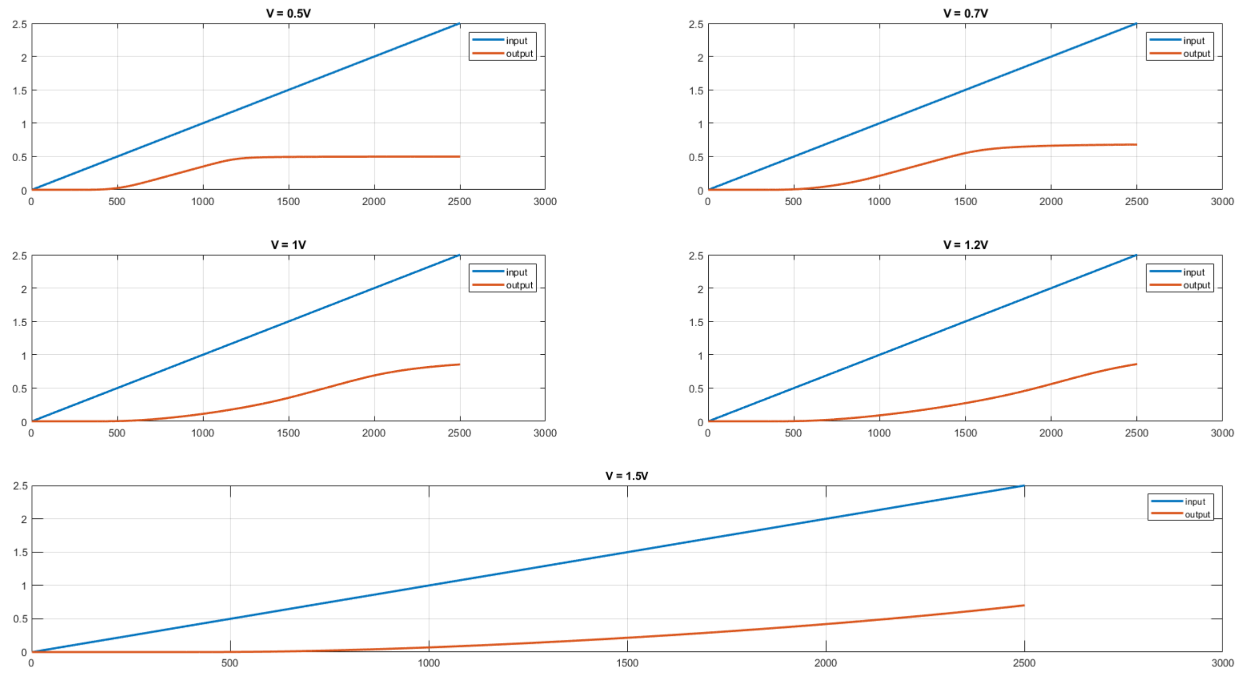

Figure 15: Testing the influence of Vr in simulation

After testing the neuron in simulation, we tested it by doing off-chip experiments.
The design of the neuron can be seen on figure 16

To match the design of the neuron previously seen, we use the following wiring for the IO PAD:

- IO PAD 1 is the input signal
- IO PAD 2 is V_r

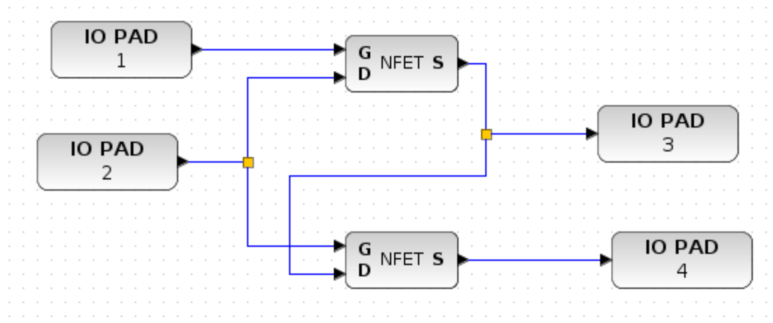

Figure 16: Scheme of the neuron

- IO PAD 3 is wired to an oscilloscope to visualize the output signal
- IO PAD 4 is wired to the mass

Again, we tried several values of V_r (0.5V, 0.7V, 1.2V, 1.5V) with a ramp in input. First we set the frequency of the ramp at a low value, we will discuss the effects of frequency on the neuron later.
We obtained the results shown on figure 17. We can see the differences between simulation and off-chip results.

The results are very different, but we still obtain a non-linear output, which is what we are looking for. We don't know why the results are so different. One can say that the simulation uses a perfect model and it doesn't represent the accurate behavior of the board. Anyway, the non-linearity is still satisfying and we can work on it. It seems even more convenient as an activation function because the saturation voltage is higher and non-linearity is more pronounced.
We chose to use the non-linearity created with $V_r = 1.5V$ for the reservoir computing experiments, since it is the one that has the greatest amplitude.

We can now observe the effect of the frequency of the input signal on the output signal. In order to do that, we used the non-linearity produced with $V_r = 1.2V$. Indeed, it is convenient since it reaches saturation, so we can clearly see the capacitive effects of the circuit on the neuron.
On figure 18, we can see the effects of the frequency of the input saw-tooth

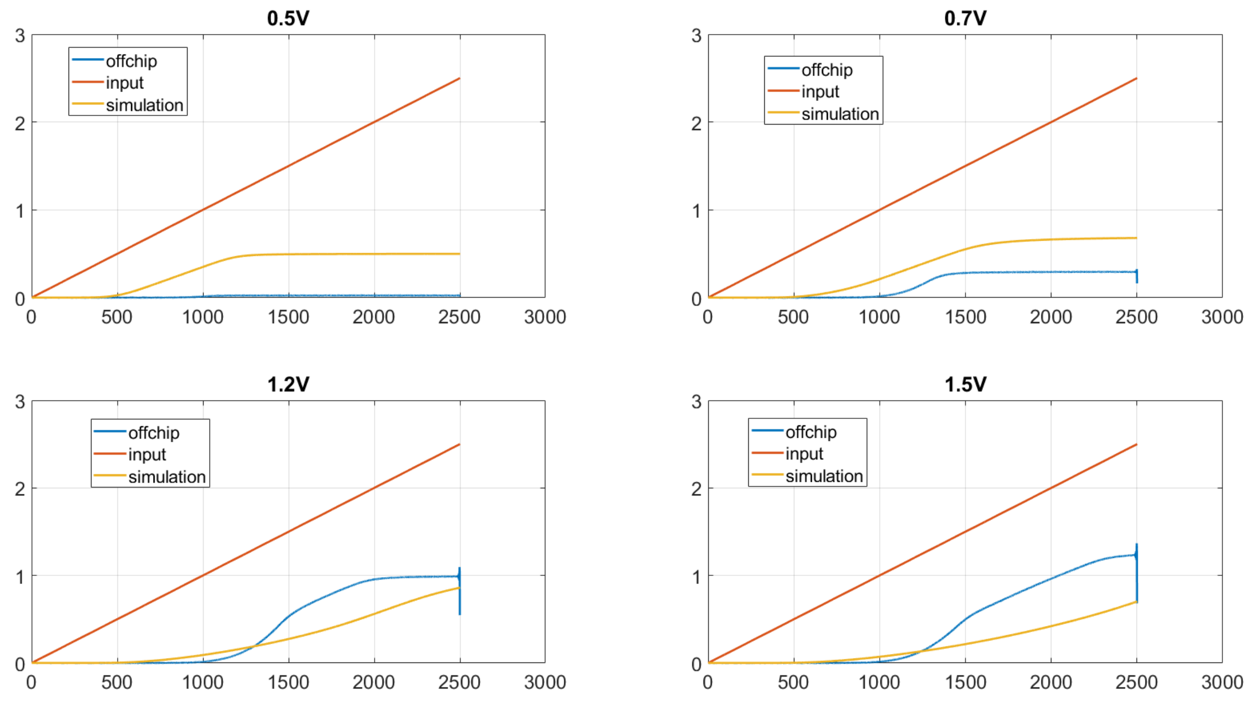

Figure 17: Comparison of the results obtained for many V_r in simulation and off-chip

signal on the output. The capacitive effects are clearly visible above about 1000Hz. So a few thousand Hz seems to be an upper limit of the input frequency.

3.2 Adding dynamic to the output : an OTA's LPF

We now have our neuron but we also want to add dynamic to the output to have an efficient reservoir.

So the idea is to add a low pass filter after the neuron. Indeed, if we choose the cut-off frequency wisely, we can have not much attenuation and a good phaseshift, which will alow us to add dynamic to the system. As we have already discussed, the idea was to implement a low pass filter using a FG transistor. We abandoned this solution since the cut-off frequency would be

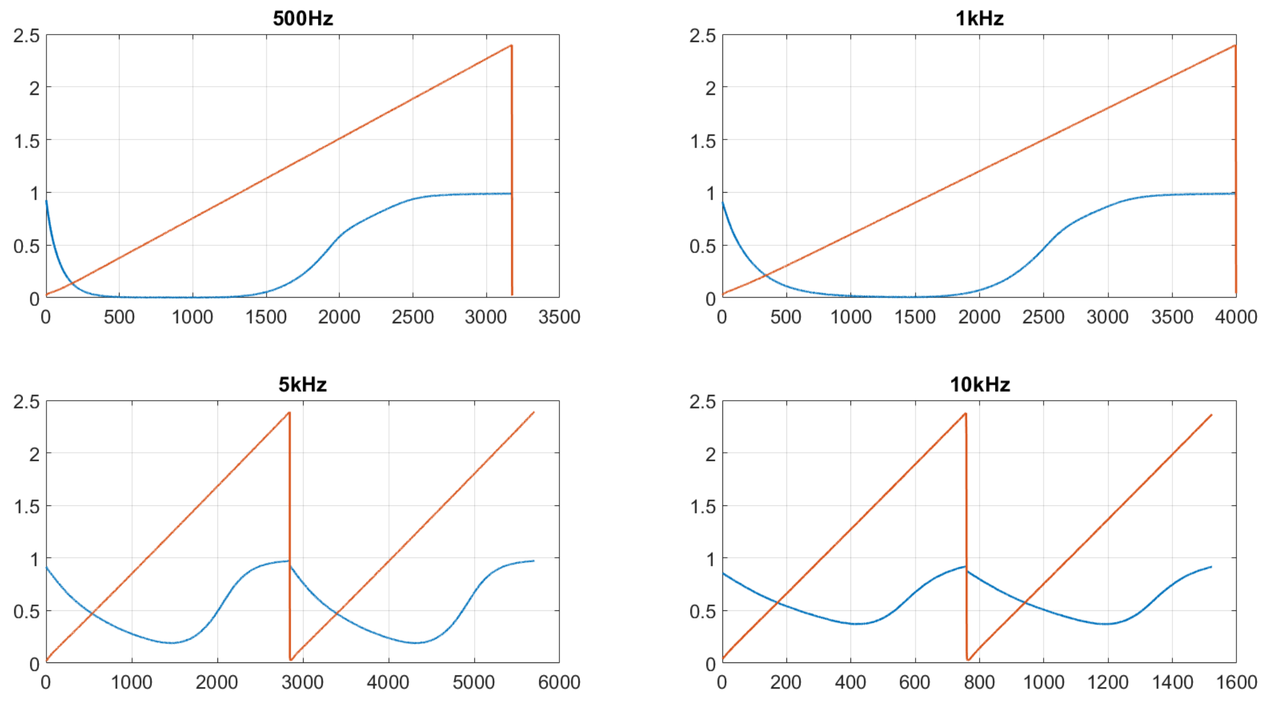

Figure 18: Influence of the input frequency on the output

too low. But, we can use an OTA to make the low-pass filter.

The scheme of the neuron with the LPF used for off-chip experiments is shown on figure 19. There is a second OTA, which is used to avoid impedance issues when we want to observe the signal on the oscilloscope.

The two N-MOS are the neuron. The first OTA is the low-pass filter. The second OTA is a buffer to make measurements with the analog discovery oscilloscope. In order to make a buffer, we choose $I_{bias} = 1e - 5A$. This allows us to make good measurements on the oscilloscope and to visualize the output signal.
The first OTA used to make a LPF is set with $I_{bias} = 5e - 9A$, which gives us a cut-off frequency of about 100 Hz.

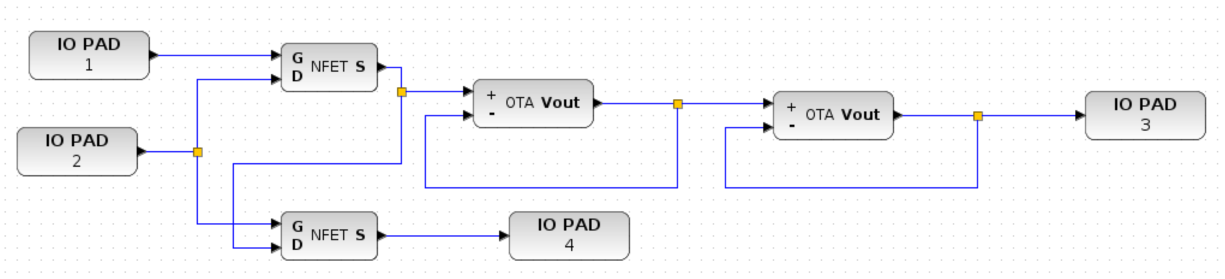

Figure 19: Scheme of the neuron with a LPF

We will observe the output of this neuron for several signals.
The response of this neuron (reminder: $V_r = 1.5V$) to a sinus of frequency 1Hz can be seen on figure 20. The signal is not attenuated and there is no phaseshift. The response to a sinus of frequency 1000Hz can be seen on figure 21. Now, we can observe an attenuation and a phaseshift.

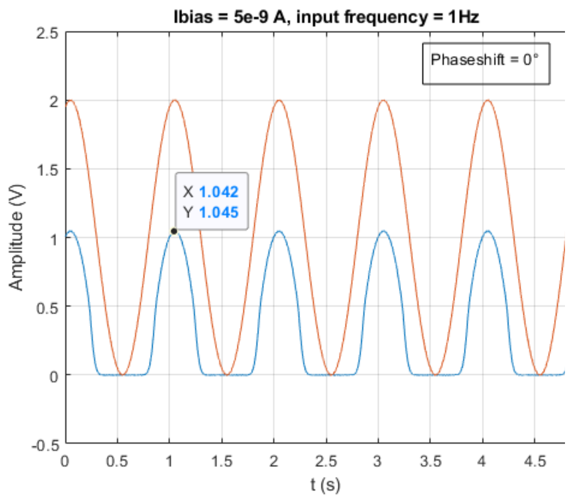

Figure 20: Response to a sinus of 1Hz

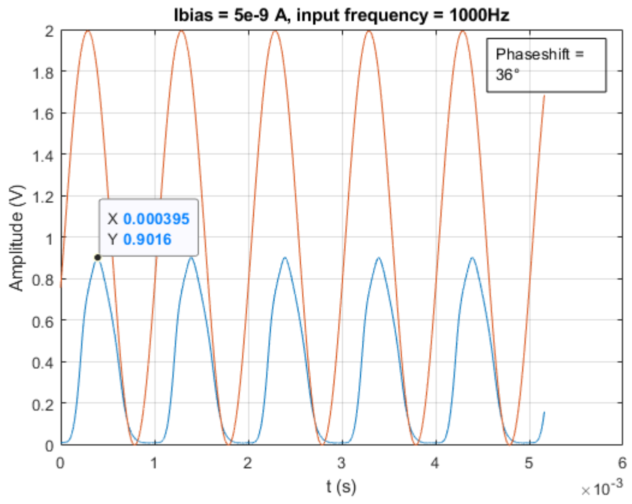

Figure 21: Response to a sinus of 1000Hz

Its step response is shown on figure 22.
As we can see, the 5% response time is about 25ms.

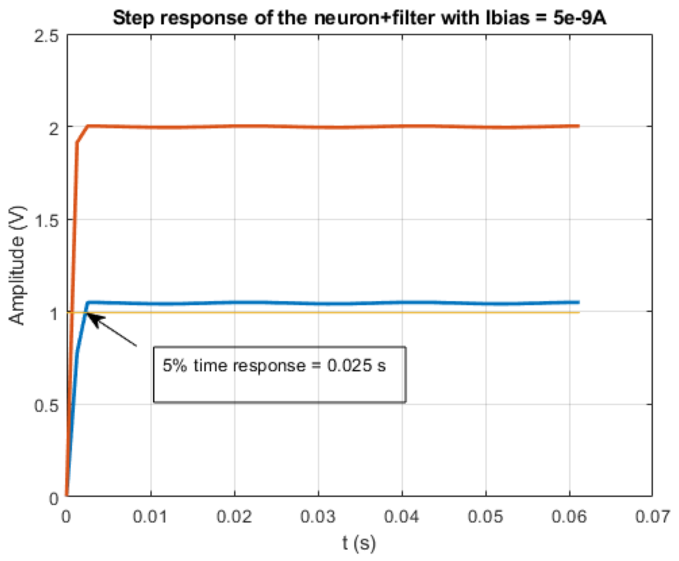

Figure 22: Step response

4 Delay based reservoir computing

4.1 Principle

Delay based reservoir computing is an implementation of reservoir computing, which uses only one non linear element (neuron). Virtual nodes are created by time-domain multiplexing: the input signal u(t) is sampled and held for a time τ while a random binary mask of length N is applied at a rate θ^{-1} with ($\tau = N\theta$). The mask defines the input weights and keeps the system from saturating by regularly switching between two levels. The functioning of the reservoir is shown on figure 23.

The input is multiplexed with the mask. It simulates a reservoir with many neurons. So it is as if the signal transited through many neurons, even if the system only uses one neuron. Then, we can collect the values obtained on each virtual node to calculate the weights of the output layer. This can be done with the least square method (this is the method we used to determine the weights of our output layer) or other methods.

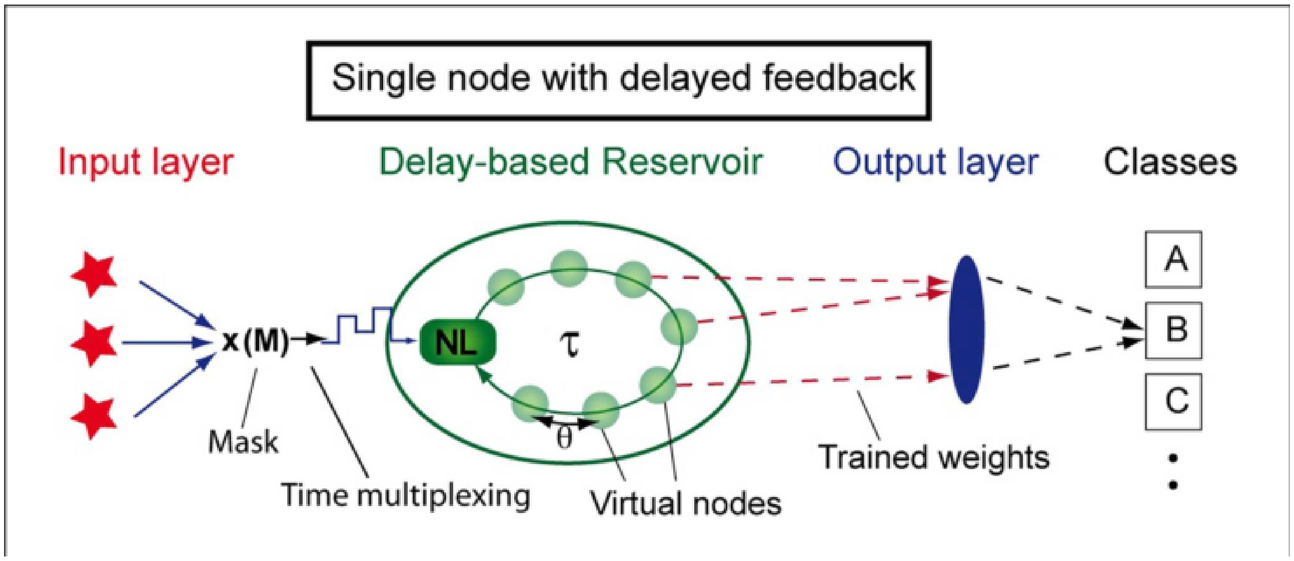

Figure 23: Principle of delay based reservoir computing

4.2 The mask

As we do not want the system to saturate, it is important for the sampling rate of the mask to be chosen wisely.

We chose to take $I_{bias} = 6e-10A$ for the OTA of the low pass filter, which gives us a cut-off frequency of about 20Hz. The response to a sinewave of 1Hz and the step response can be seen respectively on figure 24 and 25.

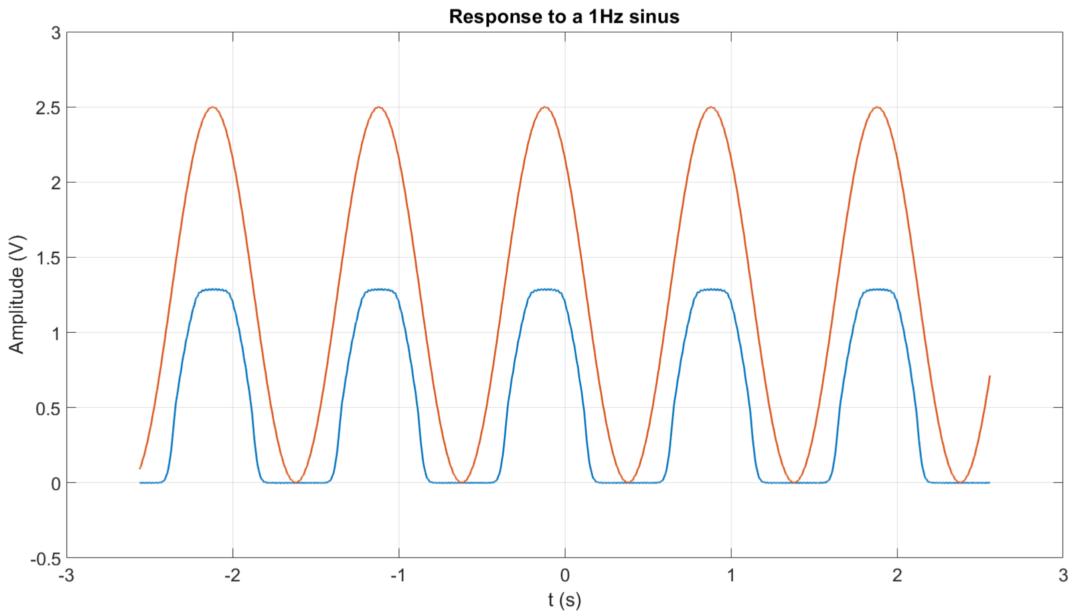

Figure 24: Response to a 1Hz sinus

Figure 26 shows us the response of the neuron to square signals with different frequencies. If the frequency is too low, the system saturates. If it is too high, we just obtain the mean value of the input signal.
As we can see, around 20Hz, the output does not saturate and is still non linear.

Now, we can observe the response of the system to the mask. We chose a sampling frequency of 200Hz for the mask. Indeed, for such a frequency, we do not observe much saturation. So it seems to be a good frequency.

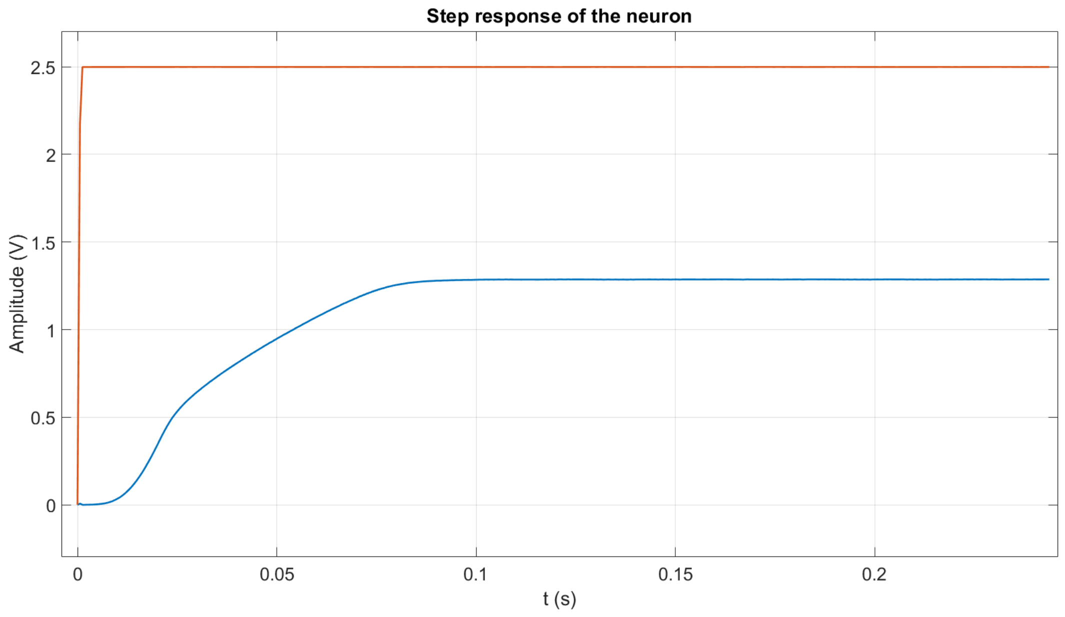

Figure 25: Step response

4.3 Decision feedback equalization (DFE) problem

The aim of this problem is to find the original signal from a noisy signal. Here, we have 5000 samples, which can take four values: -3, -1, 1 and 3. The input signal of our reservoir is a noisy version of these values. A part of the input signal and the wanted signal are shown on figure 29.
We can see that the red curve represents the wanted value, which take only four values. The blue signal is the noisy version, which takes value around -3, -1, 1 and 3. So with the reservoir, we want to classify these input values which are noisy and find the original values.

In order to implement delay based reservoir computing, we have to multiplex the noisy signal with our mask. Moreover, since the board can only work in a specific range of voltage, we modify the input signal: we normalize the signal and set its maximal amplitude to 0.1V and we add an offset of 1.25V. The process is summed up on figure 30.
We can see that the signal is multiplexed with the mask. Then we set its

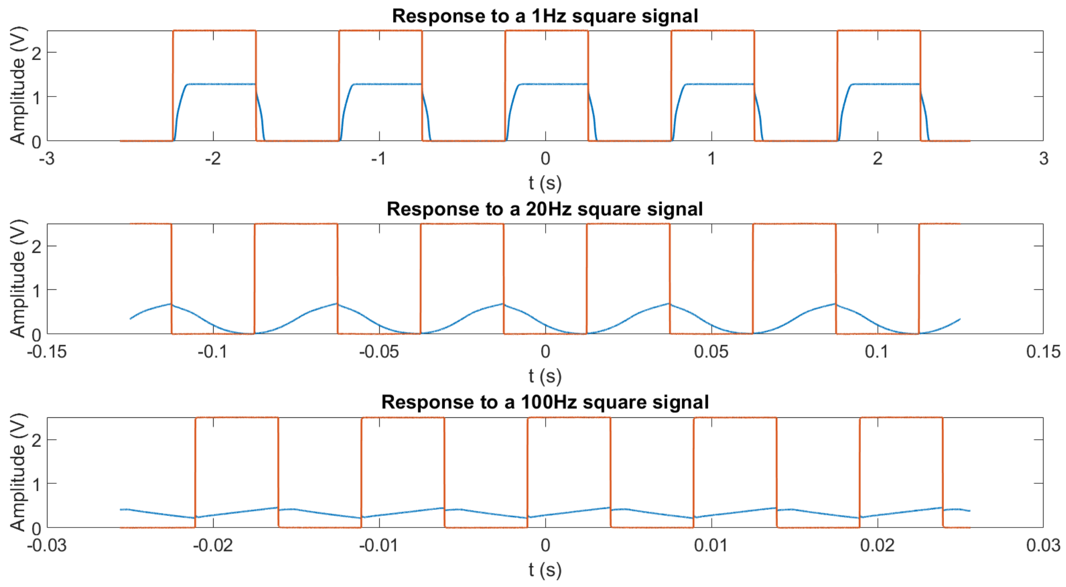

Figure 26: Influence of the frequency on the response to a square signal

maximum amplitude to 0.1 and we add an offset of 1.25V. The last step allows us to have good results. Indeed, the offset allows us to not have an output which is null or which has an amplitude too small.

Our mask allows us to have 64 virtual nodes and we will observe the output of the reservoir when we submit the input to it. We will first determine the weights of the output layer by using the least square method on a training set. This will allows us to have weights which minimizes the error between the input signal and the wanted signal. Then we will tests our weights on a testing set. For the experiments we have a frequency of 200Hz for the input signal and the frequency of the oscilloscope is 800Hz. An example of the input and output signal is shown on figure 31.

We divided the set of 5000 samples in two subsets: one of 4000 samples for the training and one of 1000 samples for the testing. We used the least square method to calculate the weights of the output layer. We obtained 96% of success rate for the training and 94% for the testing. These results

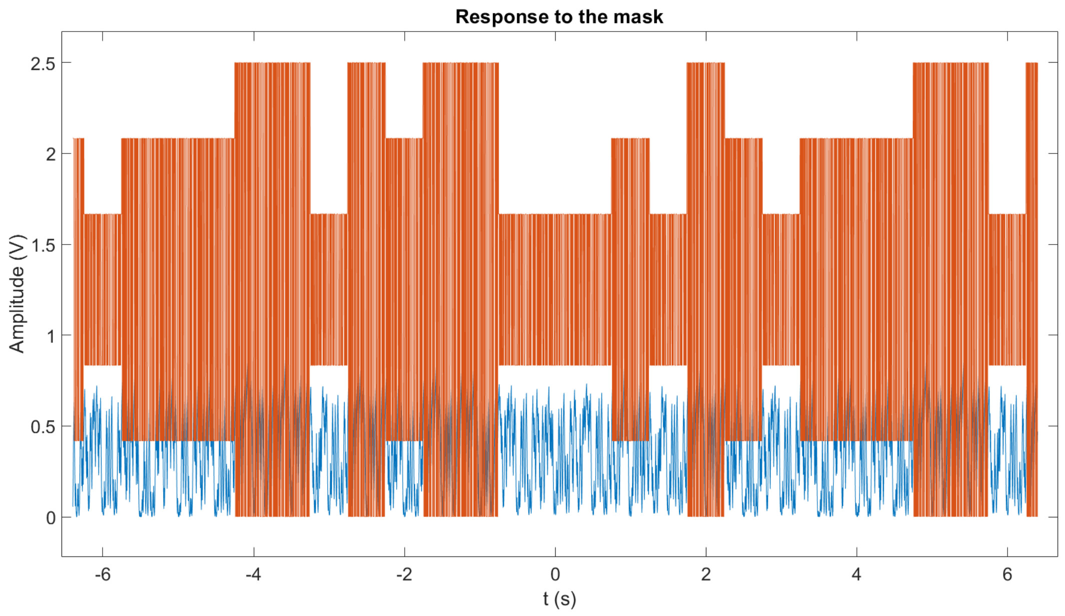

Figure 27: Response to the mask

are shown on the figures 32 and 33. The blue plot is the target signal (the signal without noise, i.e. the values -3, -1, 1, 3), the green plot is the output signal and the orange points are the values which were not put in the good category. A zoom of these results can be seen on figure 34.

These results are high and encouraging. The non linearity works good. It is normal to obtain a smaller success rate for the testing since we determined the weight of the output layer from the results obtained at the training and we use them on the testing set to check the efficiency of our reservoir.

Some further improvements could surely be made. We could test many different V_r and see the influence on the success rate. Moreover, we can surely improve the speed of the reservoir. By rising the cut-off frequency of the low pass filter, we could also rise the frequency of the input signal and train the reservoir faster. By lack of time, these experiments were not done, but it could be interesting at least to study the influence of V_r on the success rate

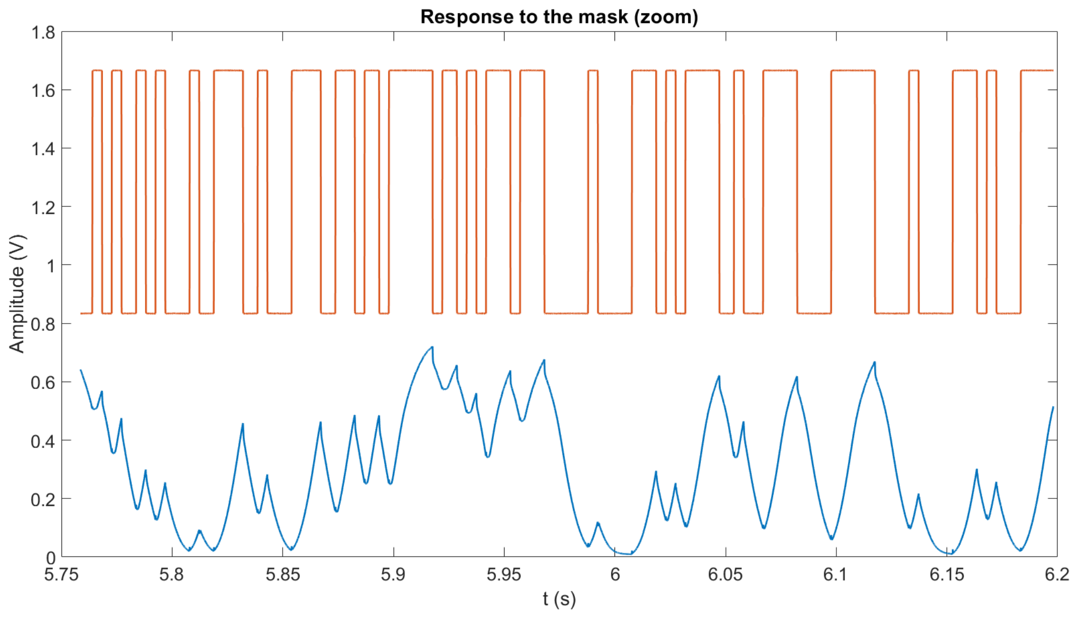

Figure 28: Zoom in on a part of the response to the mask

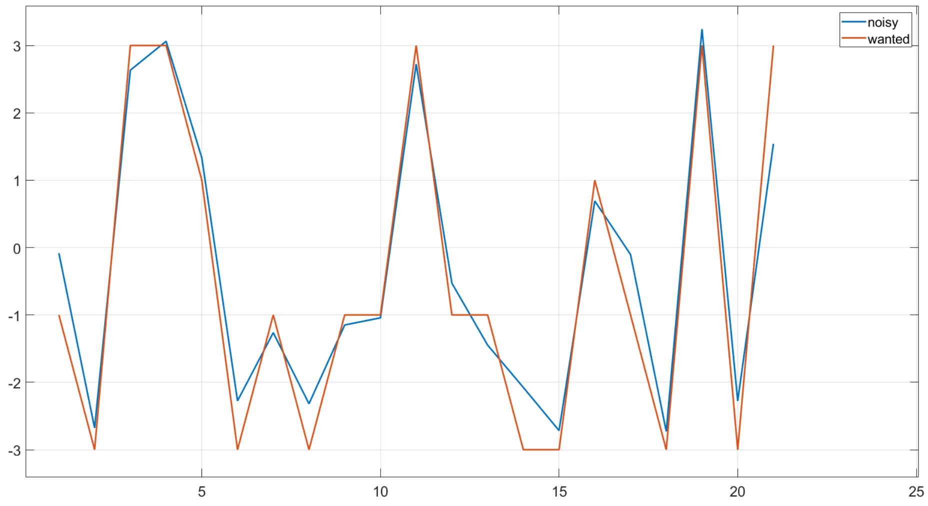

Figure 29: Input signal of the reservoir and the wanted signal

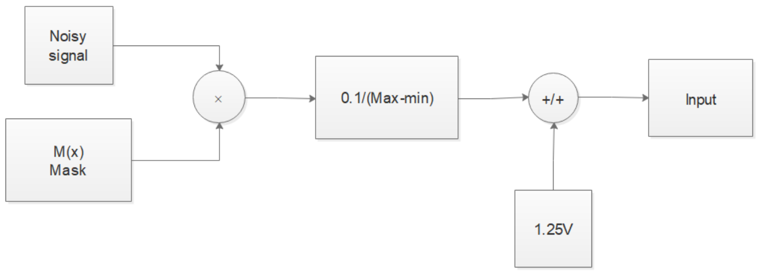

Figure 30: Making of the input of the reservoir

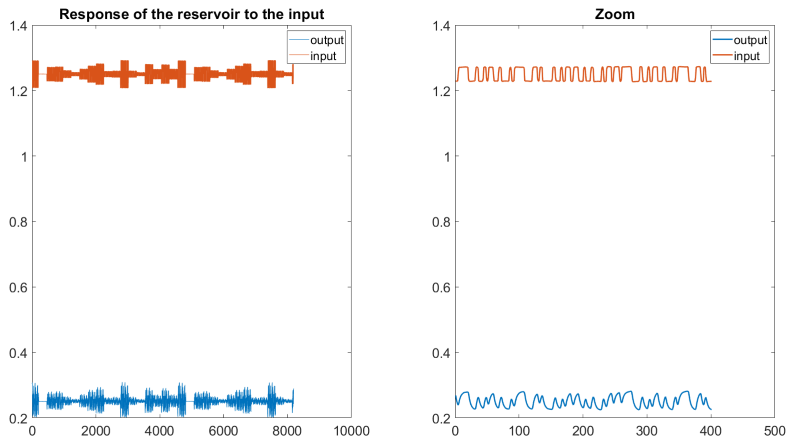

Figure 31: Response of the reservoir to the input

since it modifies the look of the non linearity.

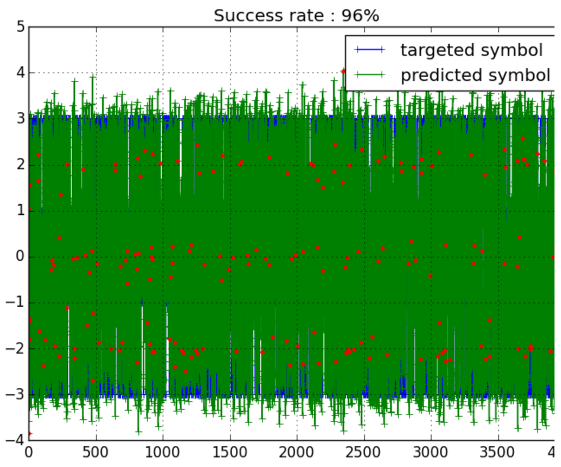

Figure 32: Training set

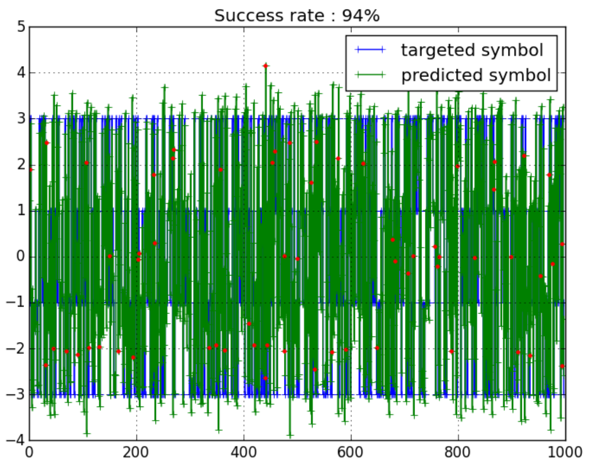

Figure 33: Testing set

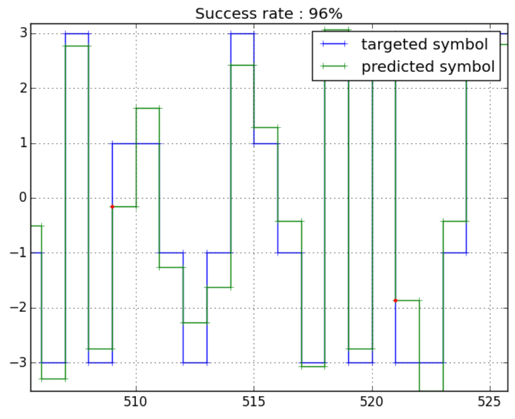

Figure 34: Response of the reservoir to the input (zoom)

Conclusion

The neuron based on the 2 N-MOS architecture seems to be promising. It has proven its efficiency on the board with the DFE problem and also off the board. Indeed, Bertrand THIA THIONG FAT and Yassine KAMRI used the non linearity produced by the neuron in their program to do spoken digit recognition. They obtained a success rate of 95% on 2500 samples.

This neuron uses less transistors than the one made by Benjamin CRITON and Hugo SCHINDLER with an OTA. Indeed, the OTA is made with 9 transistors and our neuron only uses 2 transistors. However the low pass filter are the same in the two technologies. This low pass filter may be improved. In order to use less transistors, we could try to find a solution using less than 9 transistors. This has already been tried when we wanted to implement a resistor using a floating gate transistor, but the value of the resistor was too high to produce a low pass filter with a reasonable cut off frequency.