

BEHAVIORAL MODELING FOR SAMPLING RECEIVER AND BASEBAND IN  
SOFTWARE-DEFINED RADIO

BY

SERGI ORRIT PRAT

Submitted in partial fulfillment of the  
requirements for the degree of  
Master of Science in Electrical Engineering  
in the Graduate College of the  
Illinois Institute of Technology

Approved \_\_\_\_\_  
Advisor

Chicago, Illinois  
December 2009



## ACKNOWLEDGEMENTS

I would like to thank the Illinois Institute of Technology (IIT) and Telecom UPC (Technical University of Catalonia) for giving me the opportunity to come to IIT to study a M.S. in Electrical Engineering. I would also like to thank my advisor, Dr. Yang Xu, for his advice during my completion of the Masters' Thesis.

I want to thank Fundación Vodafone for the scholarship they have offered me through their collaboration with Telecom UPC (Technical University of Catalonia), as well as the Generalitat de Catalunya (Agència de Gestió d'Estudis i de Recerca) for the scholarship they offered me. This generosity has helped greatly within this economically negative environment.

Moreover, I would like to acknowledge my family, especially my parents, Nuria Prat Ventura and Josep Orrit Estruch, whose continuous support has gotten me thus far. I also need to mention my three roommates Borja Besalduch, Álex Gonzalez, Álvaro Higuera and Nayef Alfawaz with whom I have shared my life in Chicago, my classmates Miguel Lauzurica and Nicolo Testi, and especially Kristine Carlos for supporting and helping me in way possible.

In short, I am extremely thankful for all these people who have made it possible for me to come to the Illinois Institute of Technology (IIT) and who have given me the best experience offered in order to successfully complete a M.S. in Electrical Engineering.

Sergi Orrit Prat

# TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS .....	iii
LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
ABSTRACT .....	xi
CHAPTER	
1. INTRODUCTION .....	1
2. HISTORICAL REVIEW .....	3
3. DISCRETE-TIME SIGNAL PROCESSING TOOLS .....	8
3.1 Sampler .....	8
3.2 Anti-aliasing filter .....	12
3.2.1 Sinc filter .....	13
3.2.2 Sinc <sup>3</sup> filter .....	16
3.3 Down-sampler .....	18
3.4 Up-sampler .....	21
3.5 Decimation and Interpolation .....	22
3.5.1 Decimation .....	23
3.5.2 Interpolation .....	24
3.6 Multiple Stages .....	25
3.6.1 When can multiple stages decimation be used? .....	25
3.6.2 Why is using multiple stages interesting? .....	25
3.6.3 How can the optimum number of stages and the decimation factor of each stage be chosen? .....	26
4. SAMPLING RECEIVERS .....	27
4.1 Sub-sampling receiver [7] .....	27
4.2 Direct-sampling receiver [10] .....	31
4.2.1 Multi-Tap Direct Sampling Mixer (MTDSM) .....	33
4.2.2 Sigma-Delta ADC with a built-in anti-aliasing filter [3] .....	37
4.3 Comparison between sub-sampling and direct-sampling receivers explained .....	38
4.3.1 Anti-aliasing filter .....	38
4.3.2 Down-sampling factor .....	39
4.3.3 Frequencies used .....	39

CHAPTER	Page
4.3.4 General Advantages of Direct Conversion.....	40
4.4 Simulink Model .....	40
4.4.1 Sinc filter (N=8) .....	41
4.4.2 Sinc filter (N=4) .....	41
4.4.3 Sinc <sup>3</sup> filter (N=2).....	42
4.4.4 MTDSM .....	42
4.4.5 Receiver.....	43
4.5 Results .....	46
4.6 Conclusions.....	50
5. A RECONFIGURABLE ANALOG BASEBAND FILTER .....	51
5.1 Introduction.....	51
5.2 Understanding the filter .....	56
5.2.1 Butterworth.....	59
5.2.2 Chebyshev .....	60
5.2.3 Elliptic .....	61
5.3 Verilog-A (Cadence) Model .....	63
5.3.1 Transconductor.....	63
5.3.2 Anti-aliasing filter .....	65
5.3.3 Integrator .....	73
5.3.4 DT-LPF .....	76
5.4 Results.....	78
5.5 Conclusions.....	88
6. CONCLUSIONS .....	89
APPENDIX A: Matlab code to obtain the MTDSM impulse response .....	90
BIBLIOGRAPHY .....	95

## LIST OF TABLES

Table	Page
5.1 Variables configuration for different types of filters.....	78
5.2 Cadence simulations for the 4-tap FIR-Gm filter and the Anti-aliasing filter	80
5.3 Error introduced by the 4-tap FIR-Gm filter .....	81
5.4 2 <sup>nd</sup> -order DT-LPF Cadence simulations for different configurations .....	83
5.5 2 <sup>nd</sup> -order DT-LPF error for different configurations.....	84

## LIST OF FIGURES

Figure		Page
2.1	Block Diagram of Heterodyne Architecture .....	4
2.2	Block Diagram of Superheterodyne Architecture.....	5
2.3	Block Diagram of Homodyne Architecture .....	5
2.4	Block Diagram of a Receiver employing Discrete-Time tools.....	6
3.1	Effects of the Sampler in the time domain.....	11
3.2	Effects of the Sampler in the frequency domain.....	12
3.3	Sinc Filter impulse response .....	14
3.4	Sinc Filter frequency response.....	14
3.5	Example of the Sinc filter effect with a rectangular signal.....	15
3.6	Sinc <sup>3</sup> Filter impulse response.....	17
3.7	Sinc <sup>3</sup> Filter frequency response .....	17
3.8	Example of the Sinc <sup>3</sup> filter effect with a rectangular signal .....	18
3.9	Example of down-conversion (N=3) [5].....	19
3.10	Symbol for Down-conversion .....	19
3.11	Simulation displaying the effect of down-converting.....	21
3.12.	Example of decimating .....	23
3.13	Block diagram of the Decimation operation.....	24
3.14	Block diagram of the Interpolation operation.....	24
4.1	Block Diagram of the Discrete-time Signal Processing used in the Sub-sampling receiver [7].....	27
4.2	Effects of the sub-sampling operation in the frequency domain .....	28
4.3	Effects of the decimation in the frequency domain .....	29

Figure		Page
4.4	Biquadratic filter frequency response [7] .....	30
4.5	Block Diagram of the All-Digital receiver for Bluetooth Radio [10].....	31
4.6	Effects of direct-sampling in the frequency domain.....	32
4.7	Effects of decimation in the frequency domain .....	32
4.8	Complete transistor level of the MTDSM [10].....	34
4.9	First step of the MTDSM operation [10] .....	35
4.10	Third step of the MTDSM operation [10].....	35
4.11	Fourth step of the MTDSM operation [10].....	35
4.12	First and second IIR filters frequency response [10] .....	36
4.13	Sigma-Delta ADC with a built-in anti-aliasing filter: Block diagram [3]	37
4.14	Left: Biquad filter frequency response [7]; Right: MTDSM combined frequency response of the first and second IIR filter [10] .....	39
4.15	Left: 8-Sinc filter implementation; Right: Frequency Response .....	41
4.16	Left: 4-Sinc filter implementation; Right: Frequency Response .....	41
4.17	Left: 2-Sinc <sup>3</sup> filter implementation; Right: Frequency Response.....	42
4.18	MTDSM Simulink block diagram .....	43
4.19	Receiver Block Diagram (Simulink) .....	45
4.20	Frequency response of each filter included within MTDSM .....	46
4.21	MTDSM frequency response .....	47
4.22	Simulation results_1.....	48
4.23	Simulation results_2.....	49
5.1	Block diagram of the analog reconfigurable filter [2] .....	51
5.2	Variable duty-cycle pulse generator [2].....	52



Figure		Page
5.3	4-tap FIR-Gm filter [2] .....	53
5.4	Duty-cycle controlled Discrete-time Transconductor [2].....	55
5.5	Duty-cycle controlled Discrete-time Low-pass filter [2].....	55
5.6	Functional Block diagram of the DT-LPF .....	56
5.7	Generic diagram used for finding partial transfer functions.....	57
5.8	Transconductor in Cadence.....	63
5.9	Schematic used for Transconductor simulation.....	64
5.10	Transient response of the Transconductor .....	65
5.11	4-tap FIR-Gm filter in Cadence .....	67
5.12	Control signals for the FIR filter.....	68
5.13	Transient response of the FIR filter .....	69
5.14	Anti-aliasing filter Simulink model .....	69
5.15	Passive filter for anti-aliasing .....	70
5.16	AC response of the passive filter_1 .....	70
5.17	AC response of the passive filter_2 .....	71
5.18	Anti-aliasing filter in Cadence .....	72
5.19	Schematic used for Integrator simulation .....	73
5.20	AC response of the Integrator .....	74
5.21	Inverting Integrator .....	75
5.22	Non-inverting Integrator .....	75
5.23	2 <sup>nd</sup> -order Discrete-time Low-pass filter model in Cadence .....	77
5.24	Frequency response of the Anti-aliasing filter.....	79
5.25	Cadence simulation of the 4-tap FIR-Gm filter .....	79

Figure		Page
5.26	Cadence simulation of the Anti-aliasing filter .....	82
5.27	Comparison between the DT-LPF Cadence model and the ideal response (Butterworth) (1) .....	85
5.28	Comparison between the DT-LPF Cadence model and the ideal response (Butterworth) (2) .....	85
5.29	Comparison between the DT-LPF Cadence model and the ideal response (Chebyshev) (1) .....	86
5.30	Comparison between the DT-LPF Cadence model and the ideal response (Chebyshev) (2) .....	86
5.31	Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (1) .....	87
5.32	Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (2) .....	87
5.33	Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (3) .....	88

## ABSTRACT

Software Defined-Radio (SDR) consists of a wireless communication in which the transmitter and the receiver are controlled by means of software. Its ultimate goal is to provide a single universal radio transceiver capable of multi-mode multi-standard wireless communications. Modeling of the proper circuits and new designs aimed at SDR is necessary for further development and experimentation. It sharpens our understanding of fundamental processes, helps to make decisions and provides a guide for training exercises. Due to the lack of these models two independent and different models have been created based on new proposed designs. Each modeled design belongs to a different layer of abstraction and therefore, the tool used is different as well.

The first proposed model consist of a Simulink (Matlab) file which models the discrete-time signal processing used in a Discrete-time receiver for Bluetooth Radio. The results show good performance when processing a signal that has been transmitted through a noisy channel. The signal at each step is visualized to see the individual effect of each building block.

The second proposed model narrows down the topic and focuses on a Widely-tunable, Reconfigurable Analog Baseband filter, for which a Verilog-A model, by using Cadence, has been created. The outstanding feature of the filter is that its programmability is based on the duty-cycle of the input control signals. Moreover, Verilog-A modules bring the design really close to the real circuit, allowing the designer to face problems that the real circuit will present and easing the replacement of the building blocks with new ones when desired. The results for this model show a very little

error within the passband of the filter that increases when the attenuation introduced for the stopband becomes higher.

## CHAPTER 1

### INTRODUCTION

Today, new digital architectures' main purpose is to bring digitalization as close as possible to the antenna while at the same time eliminating unnecessary analog and off-chip components. Reasoning behind this comes from the tendency telecommunication companies have—motivated by customer demand—to include every application and any last improvement into the latest modern gadgets. Because including all the hardware necessary for each desired application into a small device is unfeasible, one solution is to have a reconfigurable device using software. One such device which shares this same goal is the so called “Software-Defined Radio”.

Software implies programmability, and in the specific case of sampling receivers, it means the ability to reconfigure the components of a receiver. Instead of having analog mixers or filters, software controls generic electronics. Software introduces the capability to change some parameters without the need of either an insertion of hardware or a reset of the system, which in turn, opens the doors to build transceivers able to receive and process multi-mode multi-standard wireless communications. A device with these characteristics is the solution for a market whose demand is eager for finding a gadget that performs as many functions as possible. Software-Defined Radio is also the basis for the “Cognitive Radio”.

Cognitive Radio is the concept of a receiver that can actively monitor several environmental factors in order to detect what part of the spectrum is being utilized by licensed and unlicensed users. The receiver then transmits using the unused spectrum. Nowadays, Cognitive Radio's possible applications are innumerable.

A lot of investigation has already been done in the field that belongs to Software-Defined Radio. Some investigation is focused on tunable receivers so that they are able to reprogram all the parameters depending on the input's frequency and modulation.

Modeling provides a guide for scenario development, enhanced communication, better planning, reduced risk, and reduced costs. Therefore, the work done in this paper has been creating models for new proposed circuits which help to verify their basic functionality and validate their usefulness under specific conditions. Furthermore, it establishes a tool for further related studies. In Chapter 2 a historical review is done to provide perspective of the classic methods compared to the sampling receivers. In Chapter 3 the main Discrete-time Signal Processing tools are explained to convey to the reader an insight of the building blocks that are used in the following sections. In Chapter 4 two different receivers (from the papers "A 900-MHz RF Front-End with Integrated Discrete-Time Filtering" [7] and "All-Digital TX Frequency Synthesizer and Discrete-Time Receiver for Bluetooth Radio in 130-nm CMOS," [10]) are described and compared. The second one is also implemented in an ideal Simulink model and simulated. The results obtained show its effectiveness. The conclusions corresponding to this part are also presented at the end of Chapter 4. Chapter 5 gets closer to the circuit topology and presents a Verilog-A model of a tunable filter for SDR ("Wide-Tunable, Reconfigurable CMOS Analog Baseband IC for Software-Defined Radio" [2]). It displays the simulations of each part comparing the results to the expected behavior. It also presents the conclusions regarding Chapter 5. Chapter 6 concludes the thesis looking back at all the work done.

## CHAPTER 2

### HISTORICAL REVIEW

Last decade, classic RF receivers have been under study due to the huge increase in demand for wireless devices, most of which are portable, what implies an interest in reducing the size and the power consumption of all components. An increasing demand can also be seen in the capabilities of these gadgets, for example cell phones, that are increasingly required to include everything in a tiny phone. If consumers wanted to have multiple applications using conventional techniques, companies would need to build a bigger receiver, and even that would not work in many cases. This concept is an important drawback of classical techniques.

The solution to these problems for the RF receivers is to bring the digitalization closer to the antenna to eliminate analog components—sometimes bigger and off-chip—and hence to decrease the size and improve their capabilities through software.

The next Figures show three of the most common conventional analog front-end architectures. It can be seen how all architectures use Filters, Amplifiers and Mixers; all these components are analog. A simplified block diagram of a “Heterodyne receiver” is shown in Figure 2.1. The RF signal from the antenna is first filtered by a Band-Select Filter that removes the out-of-band signals. Afterwards, it is amplified by a Low-Noise Amplifier (LNA), which also reduces noise contribution from the succeeding stages. The LNA output is next filtered by an Image-Reject Filter in order to remove the image before being down-converted to the Intermediate Frequency (IF) by the Mixer. After, a Channel-Select Filter performs channel selection at the IF, and at the end, the demodulation or detection is carried out to retrieve the desired information.

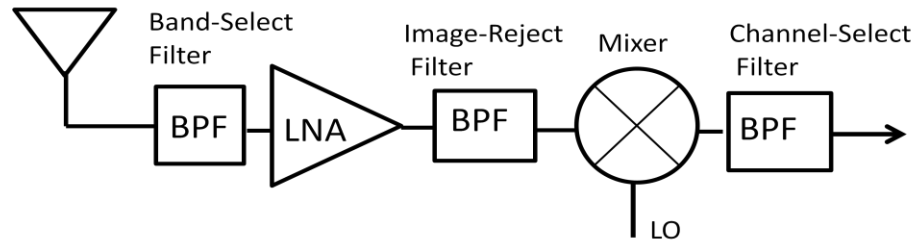


Figure 2.1. Block Diagram of Heterodyne Architecture

This single-IF scheme can lead to severe trade-offs between sensitivity and selectivity. If the intermediate frequency is high, the image appears far away from the desired signal band and can easily be suppressed by a Bandpass Filter (BPF) with typical cutoff characteristics. However, the Channel-Selection Filter would require a very high Q-factor (ratio of the center frequency to the 3dB bandwidth), and these kinds of filters are difficult to design. Instead, the channel selection has a more relaxed requirement if a low IF is used, however proper image suppression becomes harder to achieve. In practice, more than one IF mixer stage can be used to alleviate the conflict between sensitivity and selectivity. For example, that idea can be seen through a “Superheterodyne receiver”, shown in Figure 2.2, where the RF signal is first down-converted to a first IF, which is high enough to allow easy suppression of the image frequencies, and then is converted to a second IF to have a better channel selection.



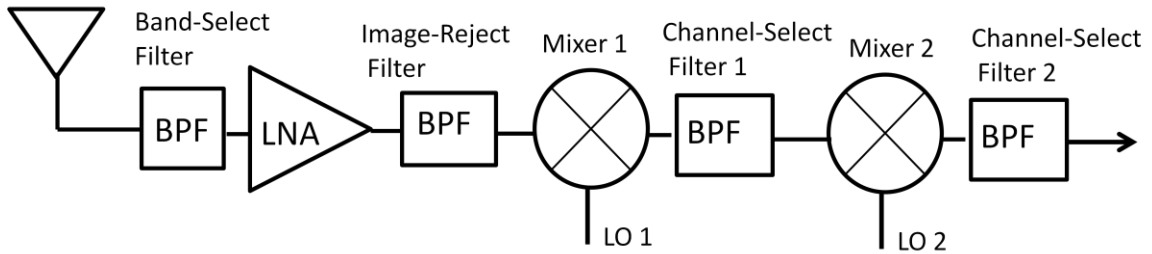


Figure 2.2. Block Diagram of Superheterodyne Architecture

In a “Homodyne or Direct-conversion receiver” (Figure 2.3), the incoming RF signal is down-converted to baseband (carrier frequency is zero) in one step by being mixed with an oscillator’s output of the same frequency. The resulting signal is then filtered with a LPF to select the desired channel.

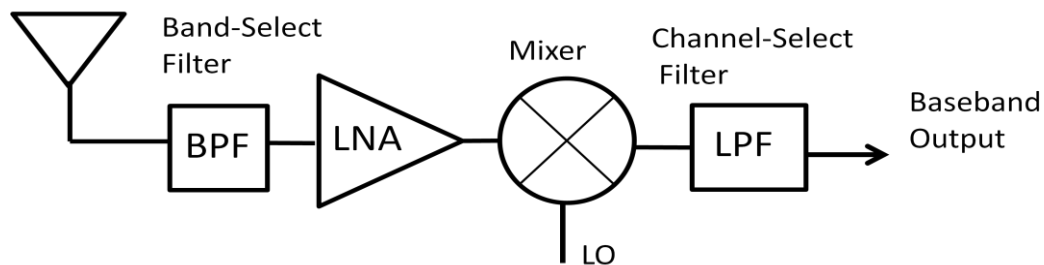


Figure 2.3. Block Diagram of Homodyne Architecture

The main advantage of a homodyne receiver is that it does not undergo the image problem because the incoming RF signal is down-converted directly to baseband without any IF stage. Another advantage of the homodyne architecture is its simplicity. Since it does not require any high frequency BPF, which is usually implemented off-chip in a superheterodyne receiver for appropriate selectivity, the homodyne receiver requires a lesser number of external components.

On the other hand, this architecture does suffer from some implementation issues. The major disadvantage is that severe DC offsets can be generated at the output of the mixer when the leakage from the local oscillator is mixed with the local oscillator signal itself. This could saturate the following stages and affect the signal detection process. Also, since the mixer output is a baseband signal, it can easily be corrupted by the large flicker noise of the mixer, especially when the incoming RF signal is weak.

Figure 2.4 shows the block diagram of a receiver employing discrete-time techniques. Its only off-chip components are the BPF and the LNA.

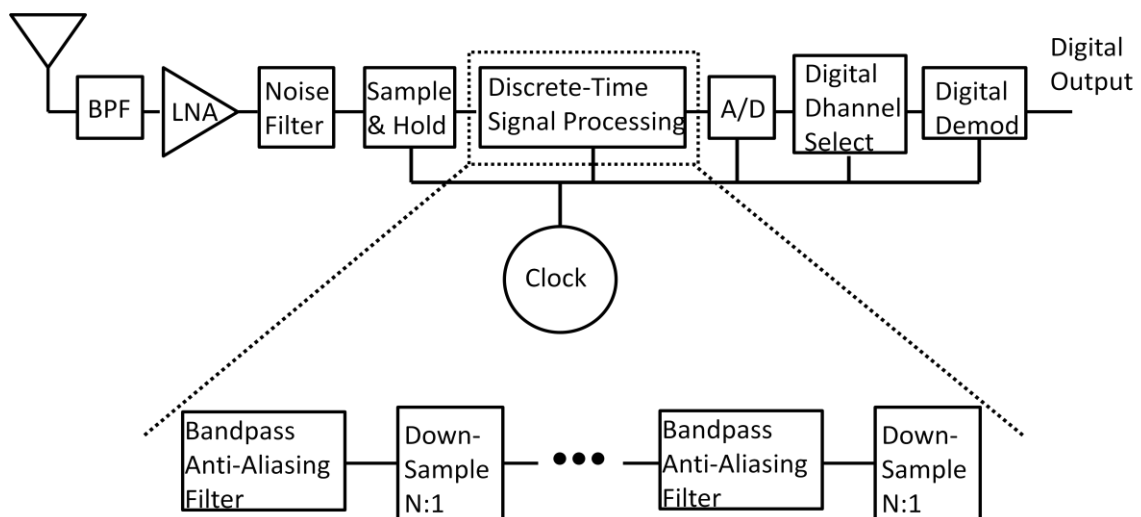


Figure 2.4. Block Diagram of a Receiver employing Discrete-Time tools

When the signal arrives at the Sample & Hold block it takes samples every period of time  $T$  ( $1/\text{Sampling Frequency}$ ). Afterwards, the signal is discrete in time and continuous in amplitude. It then applies the Discrete-Time Signal Processing techniques that are going to be explained in the next chapter. Basically, the blocks involving the processing of the input signal are cascaded pairs, each one including a Bandpass Anti-

Aliasing Filter and a Down-sampling block with decimation factor  $N$ . The reasoning behind why this receiver uses these blocks, the number of stages, and the value of the parameter  $N$  are also explained in the next chapter. The processing stages filter out the noise and, if needed, down-convert the signal to lower frequencies. The next steps are, as in the case of the classic techniques, the Analog to Digital Converter (A/D) that digitalizes the discrete-time signal converting the samples into bits, and the Digital Demodulator which takes the important information from the signal.

The following chapter explains and analyzes the Discrete-Time Signal Processing tools that are applied to the specific receivers under examination.

## CHAPTER 3

## DISCRETE-TIME SIGNAL PROCESSING TOOLS

The tools used in Multirate Digital Signal Processing, in order to treat the signal and prepare it to be demodulated, are explained next. There are four main tools: a Sampler (S&H), an Anti-aliasing Filter, a Down-converter and an Up-converter. The math involved with each tool is explained in both the time and frequency domains. The collaboration between anti-aliasing filter and down-conversion is called Decimation. If instead the up-conversion is used rather than a down-conversion, then the operation is called Interpolation.

### 3.1 Sampler

The sampler, as its name states, takes samples of a time-continuous signal every period of time  $T$ . Uniformly sampling the continuous-time signal  $x_c(t)$  every  $T$  seconds, yields the sequence  $\{x[n]\}$  given by the Equation 3.1:

$$x[n] = x_c(nT), -\infty < n < \infty \quad 3.1$$

where  $T$  is the sampling period, and  $F_T = 1/T$  is the sampling frequency.

When the continuous-time signal  $x_c(t)$  is sampled, its bandwidth is limited and the recovery of the signal can uniquely be found from the discrete-time signal if the sampling frequency is properly chosen. In frequency domain, the continuous-time signal is represented by the Fourier transform as:

$$X_c(j\Omega) = \int_{-\infty}^{\infty} x_c(t) e^{-j\Omega t} dt \quad 3.2$$

where  $\Omega = 2\pi F$  is frequency in radians per second.

The next theorem explains the conditions for recovering the continuous-time signal  $x_c(t)$  from its samples:

“If a continuous-time signal  $x_c(t)$  has a band-limited Fourier transform  $X_c(j\Omega)$ , that is  $|X_c(j\Omega)| = 0$  for  $|\Omega| \geq \Omega_N = 2\pi F_N$ , then  $x_c(t)$  can be uniquely reconstructed without error from equally spaced samples  $x_c(nT)$ ,  $-\infty < n < +\infty$ , if  $F_T \geq 2F_N$ , where  $F_T = 1/T$  is the sampling frequency”. The frequency  $\Omega_N$  is called the Nyquist frequency and the frequency  $2\Omega_N$  is referred to as the Nyquist rate.

The sampling operation is called oversampling if the sampling frequency is higher than the Nyquist rate,  $\Omega_T > 2\Omega_N$ . The term undersampling is used when the sampling frequency is lower than the Nyquist rate,  $\Omega_T < 2\Omega_N$ . And finally, the signal is critically sampled when the sampling frequency is exactly equal to the Nyquist rate,  $\Omega_T = 2\Omega_N$ . The spectrum of the discrete-time signal  $X(e^{j\omega})$  can be expressed in terms of the continuous-time signal's spectrum  $X_c(j\Omega)$  as follows,

$$X(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(j\frac{\omega}{T} - j\frac{2\pi k}{T}\right)$$

3.3

The spectrum of the discrete-time signal  $X(e^{j\omega})$  is an infinite sum of shifted and scaled replicas of the spectrum of the continuous-time signal  $X_c(j\Omega)$ . In this case,  $\omega = \Omega T$ . Equation 3.3 shows that when the sampling is performed in a sufficiently high rate, the spectrum of the discrete signal appears as a periodic repetition of the original spectrum. The original signal  $x(t)$  can be established by selecting the baseband spectrum of  $X(e^{j\omega})$  by using a LPF, otherwise, the undersampling causes aliasing in the spectrum thus making the signal recovery impossible.

The reconstructed signal  $x_r(t)$ , can be expressed in terms of the discrete signal  $\{x[n]\}$  and the impulse response of the reconstruction filter  $h_r(t)$ ,

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n]h_r(t - nT)$$

3.4

If the filter was ideal, then  $x_r(t)$  would happen to be equal to  $x(t)$ , but ideal behavior it is unfeasible and so some tolerance to this conversion has to be considered.

Some Matlab® simulations have been made to graphically observe what happens to the signal. Two functions have been created:

- “data.m”: It creates a signal composed for two sinusoids at frequencies of 300 KHz and 1 MHz. It returns two signals, the one with the two sinusoids (x) and another one that also has Gaussian noise (y).
- “sampler.m”: It takes samples every period of time (number of points). As will be seen later, the sampler here is a down-converter due to Matlab® works with discrete-time signals. So instead of taking samples every period of time, it takes a sample every N input signal samples.

The Matlab® code and the Figures obtained are the following:

```
function [x,y]=data;

t = 0:.000000001:.0025;
x = sin(2*pi*300E3*t) + sin(2*pi*1000E3*t);
y = x + 2*randn(size(t)); % adds Gaussian noise

subplot(1,2,1); plot(t,x);
title('Continuous time'); xlabel('Time (s)'); ylabel('Amplitude');

function [out]=sampler(vin,N);
% In this case the sampler is just a down-sampler since the signal in Matlab® is discrete
out=zeros(floor(length(vin)/N),1);
for i=1:length(out)
```

```

    out(i)=vin(i*N);
end

subplot(1,2,2); plot(out);
title('Discrete time'); xlabel('Samples'); ylabel('Amplitude');

```

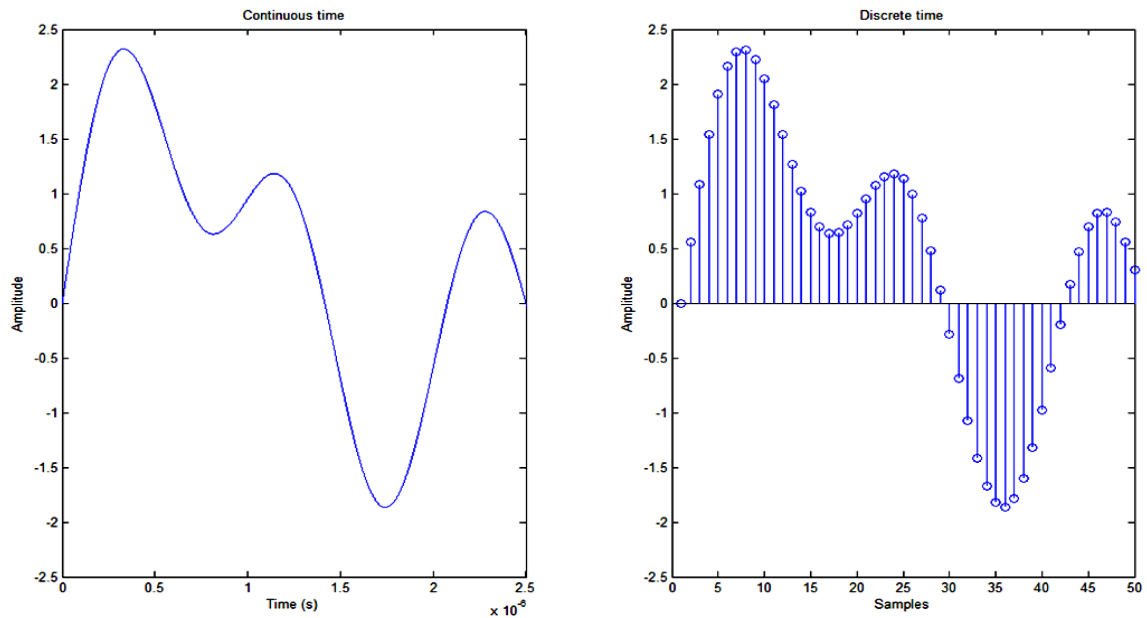


Figure 3.1. Effects of the Sampler in the time domain

The function of a Sampler is showed in Figure 3.1, where the continuous signal has been represented in a lineal form so that the real effect of sampling is observed.

If the frequency domain is plotted (See Fig. 3.2) replicas of the signal appearing at frequencies multiple of  $F_s$  (sampling frequency) can be seen. Here, the signal has been built so that the Nyquist Theorem is accomplished. If the signal's bandwidth was, e.g., in this case four or five times higher, aliasing would appear, and hence, receivers would not be able to recover the signal without errors.

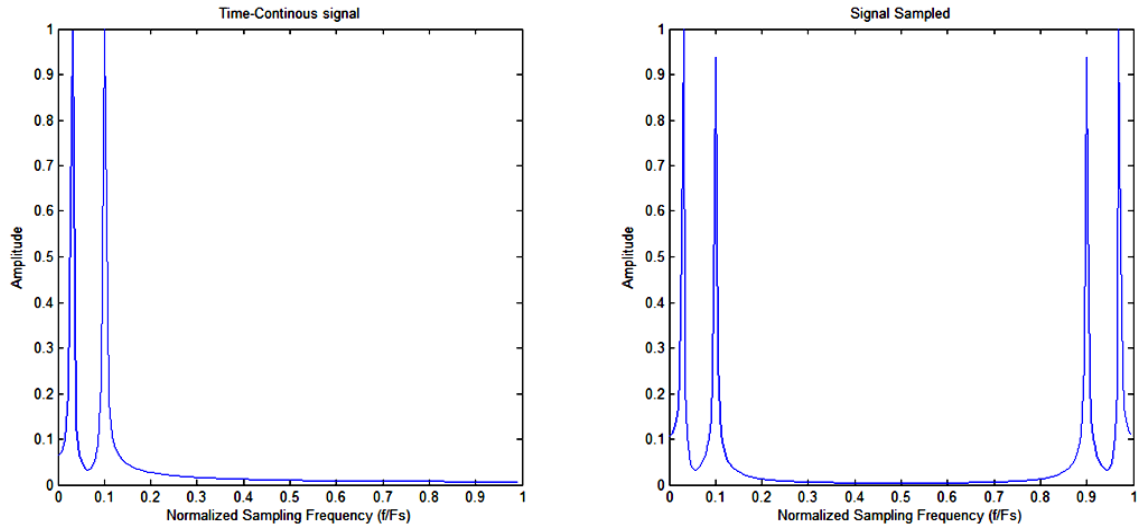


Figure 3.2. Effects of the Sampler in the frequency domain

### 3.2 Anti-aliasing filter

The Anti-aliasing filter filters the discrete-time signal preceding the down-conversion –it attenuates the frequency components outside the baseband of the signal (avoids aliasing) so that when the signal is down-converted no aliasing occurs. There is also another filter, sometimes called the Antiimaging filter, which follows the up-conversion operation where it attenuates unwanted periodic spectra which appear in the new baseband.

The anti-aliasing filter that is here studied is the Sinc Filter. The reasoning behind why this type of filter has been chosen will be explained later. Its transfer function is expressed by  $H(z)$ :

$$H(z) = \left\{ \frac{1}{N} \sum_{n=1}^{N-1} z^{-i} \right\}^M$$



where  $M$  is the filter order and  $N$  is the decimation factor . The output samples  $y$  of the filter at time  $n$ , as a function of the input samples  $x$ , can be written as:

$$y(n) = \left\{ \frac{1}{N} \sum_{n=1}^{N-1} x(n-i) \right\}^M \quad 3.6$$

At the receiver explained at part 4.2, three Sinc Filters are used, two first-order filters and one third-order filter. A low power implementation of the Sinc<sup>3</sup> Filter can be seen at [4]. The implementation and simulation of these two filters, done in Matlab®, are showed below.

### 3.2.1 Sinc Filter.

```
function [out] = sinc_filter(vin,N);
%the sinc filter is created just adding multiple delayed input samples

out=zeros(length(vin),1);
for n=N:length(vin),
    for i=0:N-1,
        out(n)=out(n)+vin(n-i);
    end
end
out=out/N;
```

Figures 3.3 and 3.4 correspond to the filter when its decimation factor  $N$  is equal to 4, the reason being that there are simply 4 samples of amplitude 1 in its impulse response and the remaining are 0. Generally speaking, there will be as many samples different of 0 (equal to 1) as  $N$ .

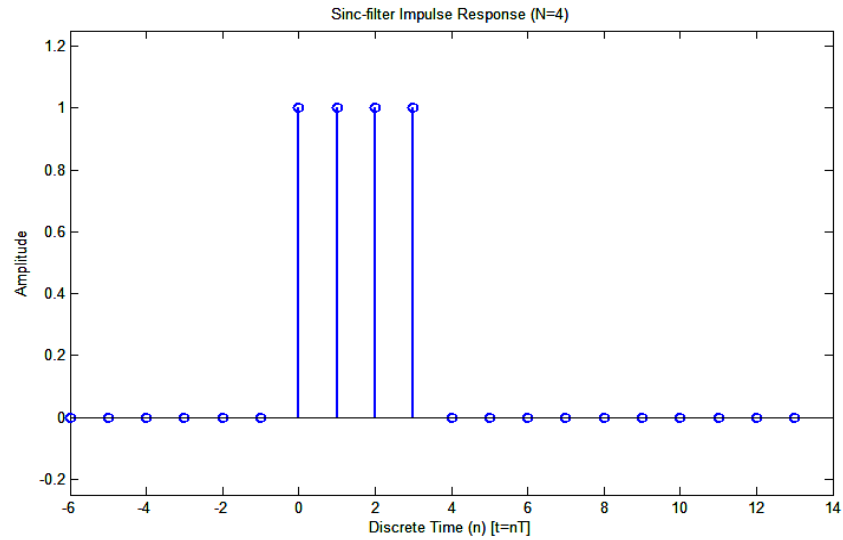


Figure 3.3. Sinc Filter impulse response

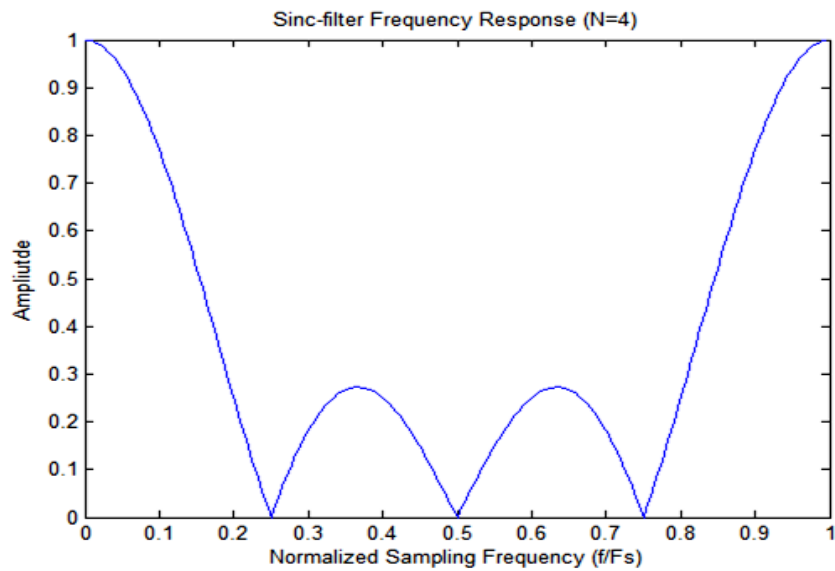


Figure 3.4. Sinc Filter frequency response

In the frequency response, Figure 3.4, it can be seen that the response is symmetric around  $F_s/2$ . The fact that  $N = 4$  implies locating three notches at multiple of frequencies  $F_s/4$ , and in general, an  $N$ -Sinc Filter will generate notches at the frequencies

$F_s/N$ .  $N$  is named Decimation Factor since, depending on  $N$  a Sinc Filter eliminates the noise at a certain frequencies. Then, once the signal is  $N-1$  down-converted, the signal is placed where the first notched had cleaned the spectrum before.

Next step is creating a simple signal and filtering it to see what the effects of this filter are. The following Matlab® code generates a rectangular signal that passes through the Sinc Filter. The solid line is the input signal, whereas the dashed one represents the output signal.

```

vin=[1 0 1 0 1 0 1 0];
rectangular=rectpulse(vin,20);
out=sinc_filter(rectangular,4);
plot(rectangular); hold on; plot(out);

```

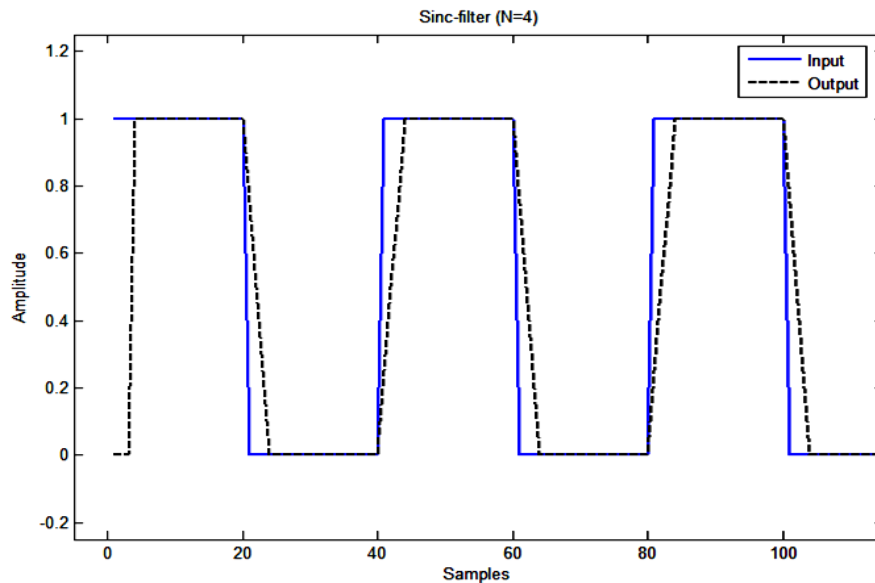


Figure 3.5. Example of the Sinc filter effect with a rectangular signal

As expected, the filter attenuates fast transitions of the signal since it is a Lowpass filter (LPF) (Figure 4.5).

### 3.2.2 Sinc<sup>3</sup> Filter.

```
function [out] = sinc3_filter(vin,N);

out=zeros(length(vin),1);
for n=N:length(vin),
    for i=0:N-1,
        out(n)=out(n)+vin(n-i);
    end
end
out=out/N;
out=out.^3;
```

In Figure 3.6,  $N$  is again equal to 4 as well. However, this filter is a third-order Sinc filter which equals to the convolution of three Sinc filters. Therefore, knowing that the number of samples  $M$  after a convolution is the sum of the size  $N_i$  of both inputs minus one:

$$M = N_1 + N_2 - 1 \tag{3.7}$$

and after a convolution of three signals, the size is,

$$K = M + N_3 - 1 = N_1 + N_2 + N_3 - 2 = \{\text{if } N_i = 4\} = 4 + 4 + 4 - 2 = 10 \tag{3.8}$$

which matches with the size of 10 samples observed in the Figure 3.6.

The frequency response of this filter is showed in Figure 3.7. Again, it has the same number of notches that the filter above had, but the lobules are much lower. It attenuates the noise better than the filter before, but at the same time, it has a reduced bandwidth.

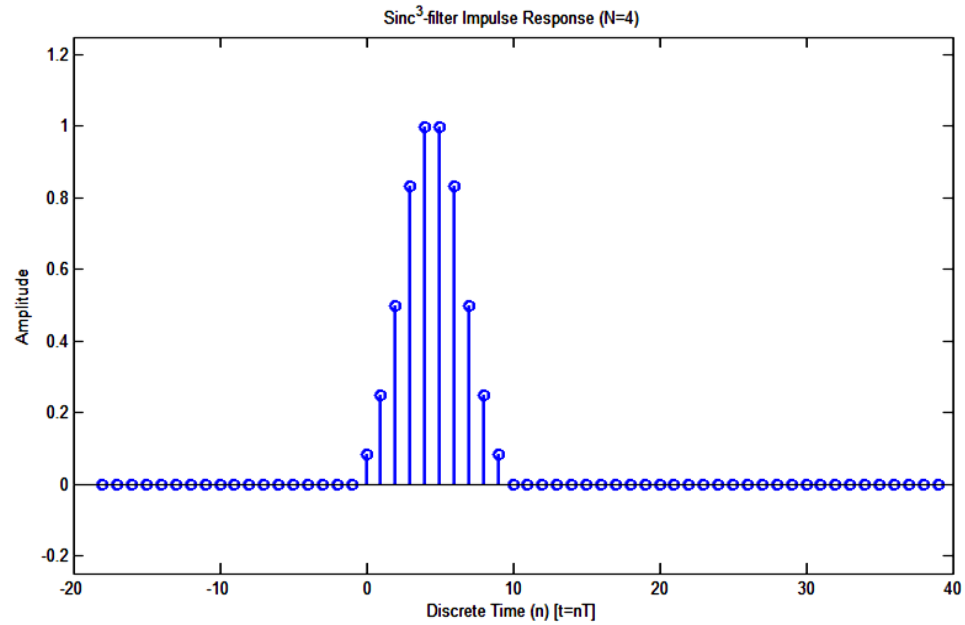


Figure 3.6. Sinc<sup>3</sup> Filter impulse response

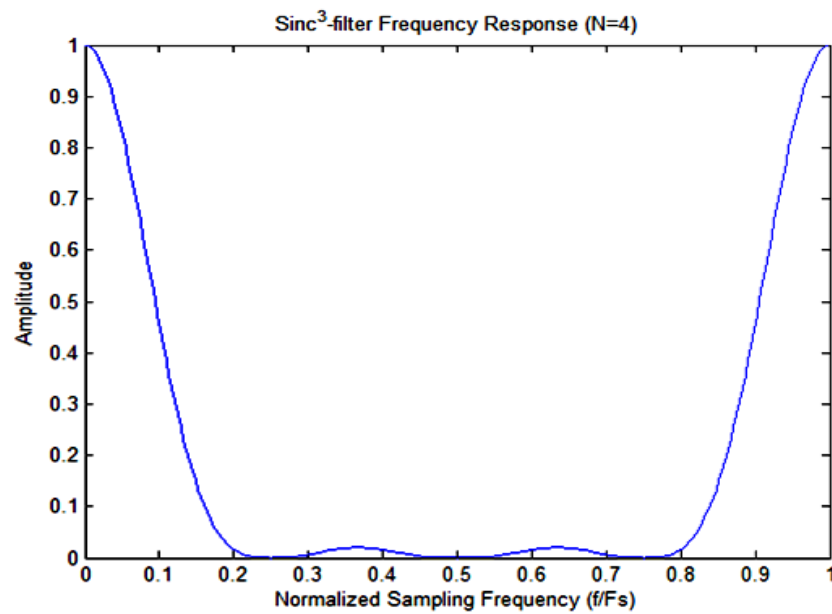


Figure 3.7. Sinc<sup>3</sup> Filter frequency response

The filter attenuates fast transitions of the signal, but in this case, the transitions are much smoother (Figure 3.8).

```

vin=[1 0 1 0 1 0 1 0];
rectangular=rectpulse(vin,20);
out=sinc3_filter(rectangular,4);
plot(rectangular);
hold on
plot(out);

```

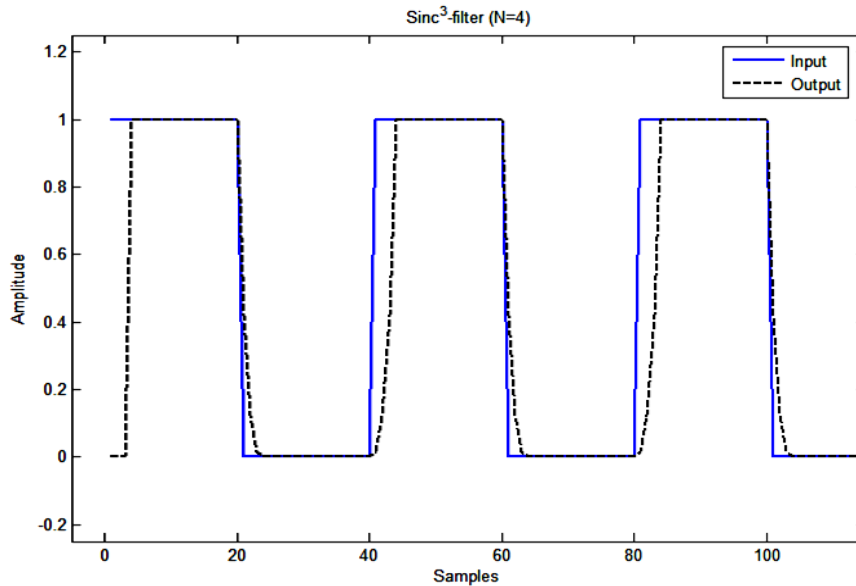


Figure 3.8. Example of the  $\text{Sinc}^3$  filter effect with a rectangular signal

### 3.3 Down-sampler

The down-sampling operation with a factor  $M$ , where  $M$  is a positive integer, is done by discharging  $M-1$  consecutive samples and retaining every  $M^{\text{th}}$  sample. Down-sampling the discrete signal  $\{x[n]\}$ , produces the down-sampled signal  $\{y[m]\}$

$$\{y[m]\} = \{x[mM]\} \quad 3.9$$

The down-sampling is the result of a two step operation. Figure 3.9 illustrates the two-step description of the down-sampling operation for a factor  $M = 3$ .

A symbol representing the down-sampling operation is shown in Figure 3.10. The box with a down pointed arrow followed with the factor M is used to symbolize the down-sampling operation.

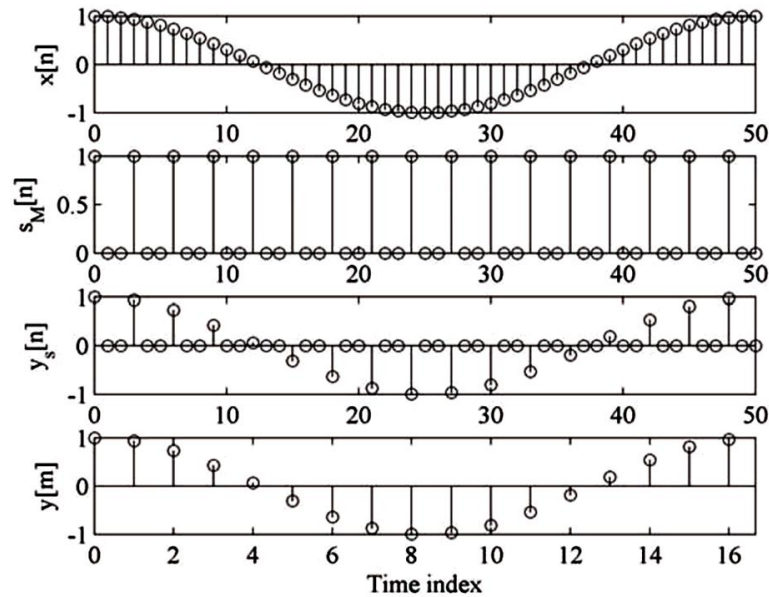


Figure 3.9. Example of down-conversion ( $N=3$ ) [5]



Figure 3.10. Symbol for Down-conversion

This operation reduces the sampling frequency  $F_T$  of the original signal  $\{x(nT)\}$ . Thus, the sampling frequency  $F_T'$  of the signal  $\{y(mT')\}$  is  $M$  times smaller than the sampling frequency of the original signal.

Frequency-domain representation of down-sampling and up-sampling is used to investigate the effects of the sampling rate alterations on the signal's spectrum. The

input-output relationship for the sampling rate alteration devices, already defined in time domain, has to be expressed in terms of z-transform and in terms of Fourier transform. This is achieved by relating the spectrum of the down-sampled/up-sampled signal with the spectrum of the original signal. We first will consider the z-domain representation of down-sampling. The input-output relationship of a down-sampler in time domain is given in Equation 3.9. Applying the z-transform to both sides of Equation 3.9, we find:

$$Y(z) = \sum_{m=-\infty}^{\infty} x[Mm]z^{-m}$$
3.10

After developing long equations the following equality can be reached, and shows the frequency relationship between input and output,

$$Y(e^{j\omega}) = \frac{1}{M} \sum_{k=0}^{M-1} X(e^{j(\omega-2\pi k)/M})$$
3.11

The above relation explains the implication of the down-sampling on the spectrum of the signal. Evidently, the spectrum  $Y(e^{j\omega})$  is a sum of  $M$  uniformly shifted and stretched versions of  $X(e^{j\omega})$  scaled by a factor  $1/M$ . Equation 3.11 shows that aliasing will occur when the bandwidth of the original signal exceeds  $\pi/M$ . Thus, only signals which are bandlimited to  $\pi/M$  can be down-sampled without distortion. For the down-sampling factor  $M$ , the highest frequency in the spectrum of  $X(e^{j\omega})$  should be limited to be less than or equal to  $\pi/M$ .

Next, a Matlab® code is displayed in order to prove graphically the effects of down-sampling.



```

function [out] = downsampler(vin,N); %vin is the output of data2.m
%it takes one sample out of N samples of the input signals

out=zeros(floor(length(vin)/N),1);
for i=1:length(out)
    out(i)=vin(i*N);
end

%Matlab® has its own function to downsample called downsample(x,r)

```

Figure 3.11 shows the down-sampling effect in the time-domain

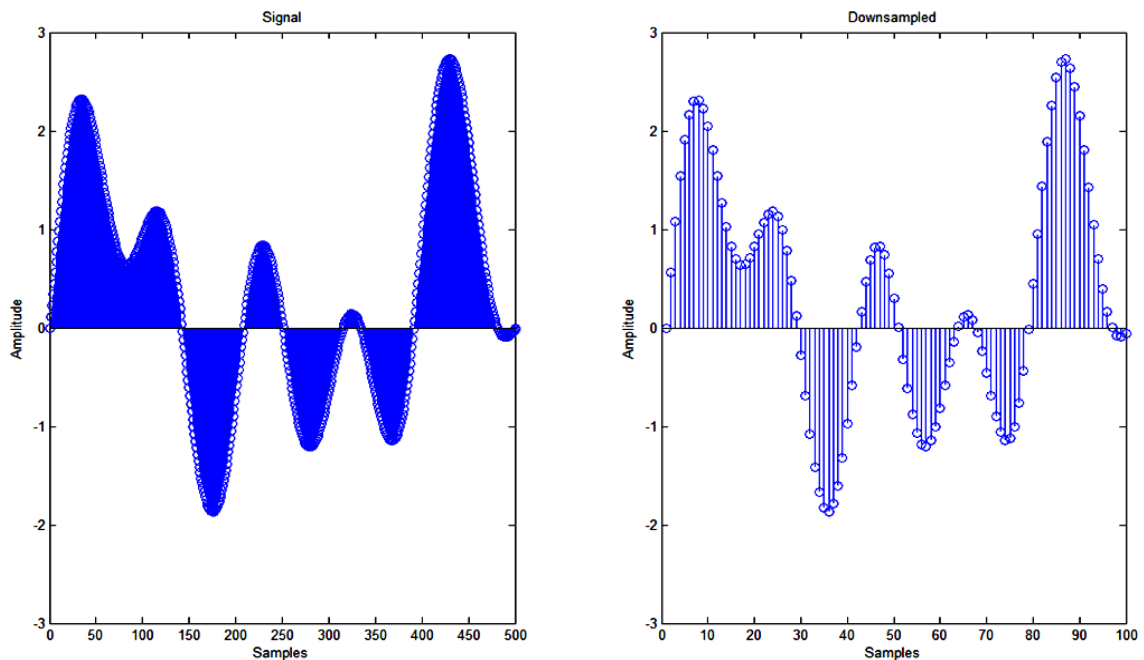


Figure 3.11. Simulation displaying the effect of down-converting

### 3.4 Up-sampler

The up-sampling operation with an integer factor  $L$  is performed by inserting  $L-1$  zeros between each pair of consecutive samples. The up-sampling operation when the input is  $\{x[n]\}$ , produces the up-sampled signal  $\{y[m]\}$  where

$$y[m] = \begin{cases} x[m/L], & m = 0, \pm L, \pm 2L, \dots \\ 0, & \text{otherwise} \end{cases}$$

3.12

The up-sampling operation increases the sampling rate  $F_T$  of the original signal  $x(nT)$ . The sampling frequency  $F_T'$  of the signal  $y(mT')$  is  $L$  times larger than the sampling rate of the original signal, i.e.,  $F_T' = LF_T$ . By definition, the  $z$ -transform of the up-sampled sequence  $\{y[m]\}$  is the following

$$Y(z) = \sum_{m=-\infty}^{\infty} x[m]z^{-Lm} = X(z^L)$$

3.13

where  $X(z)$  is the  $z$ -transform of the original signal. If  $z$  is replaced with  $e^{j\omega}$ , the frequency-domain relationship between the input and the output signals is obtained.

$$Y(e^{j\omega}) = X(e^{j\omega L})$$

3.14

Equation 3.14 shows that the factor-of- $L$  up-sampling leads to  $L$ -fold repetition of the original spectrum  $X(e^{j\omega})$  in baseband. This process is called imaging because  $L-1$  “images” of the input spectrum appear in the output.

### 3.5 Decimation and Interpolation

The process of decreasing the sampling rate is called Decimation, and the process of increasing the sampling rate is called Interpolation. The two operations previously explained, down-sampling and up-sampling, are used to change the sampling rate of the signal. The drawback of the down-sampling is the aliasing effect, whereas the up-sampling produces unwanted spectra within the frequency band of interest. Decimation must be performed in such a way as to avoid the effects of aliasing, which occurs when

the highest frequency in the spectrum of a down-sampled signal exceeds the value  $\pi/M$ . When interpolating, the  $L-1$  images caused by inserting  $L-1$  zeros between the samples should be removed.

**3.5.1 Decimation.** Decimation requires preventing aliasing. Hence, prior to down-sampling with the factor of  $M$ , the original signal has to be bandlimited to  $\pi/M$ . This means that the factor-of- $M$  decimation has to be implemented in two steps:

- (1) Bandlimiting of the original signal to  $\pi/M$
- (2) Down-sampling by the factor-of- $M$

Figure 3.12 shows an example of decimating by a factor of  $N = 2$ .

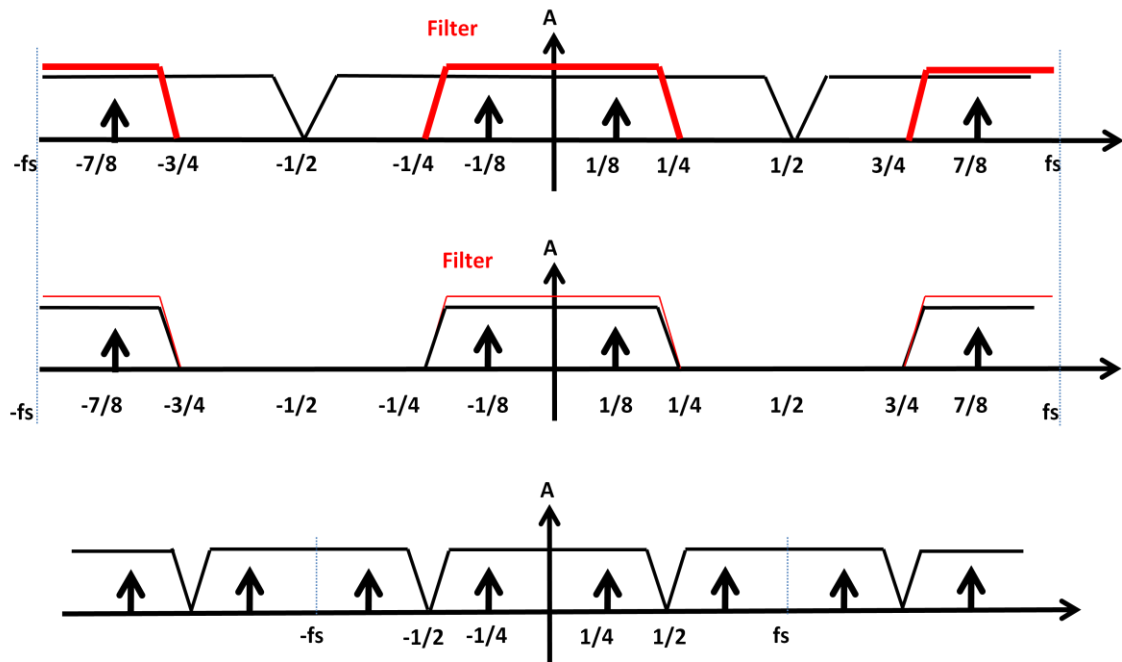


Figure 3.12. Example of decimating

The role of the decimation filter  $H(z)$  is to suppress aliasing to an acceptable value. Therefore, the performance of a decimator is mainly determined by the filter

characteristics. Since the filter with an ideal frequency response cannot be achieved, some amount of aliasing has to be tolerated.

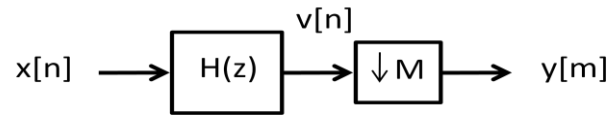


Figure 3.13. Block diagram of the Decimation operation

**3.5.2 Interpolation.** Interpolation requires the removal of the extra images created while up-sampling. This means that the factor-of-L interpolation has to be implemented in two steps:

- (1) Up-sampling of the original signal by inserting  $L-1$  zero-valued samples between two consecutive samples
- (2) Removal of the  $L-1$  images from the spectrum of the up-sampled signal.

The anti-imaging (interpolation) filter  $H(z)$  is used to remove images from the spectrum of the up-sampled signal. Removal of images from the spectrum of the signal causes the interpolation of the sample values in time domain. The zero-valued samples in the up-sampled signal  $\{x_u[m]\}$  are “filled in” with the interpolated values. As in the case of a decimator, the performance of an interpolator is mainly determined by the filter characteristics.

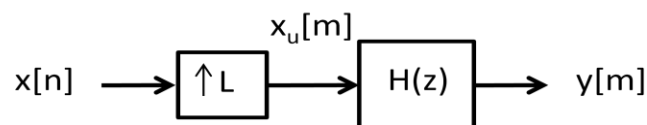


Figure 3.14. Block diagram of the Interpolation operation

### 3.6 Multiple stages

- Multistage structures are very useful for implementing large sampling-rate conversion factors. A single decimation/interpolation filter with a very narrow passband, usually inconvenient for the design and implementation, is replaced with the cascade of simpler filters. The specifications for those individual filters are significantly relaxed since the overall filter specification is shared between several lower-order filters.
- Moreover, comparing the computational efficiency between a single-stage and a double-stage decimator, the second one nearly doubles the first one. So, in other words, this technique reduces the cost of processing.

**3.6.1 When can multiple stages decimation be used?** There is one requirement which must be accomplished: The decimation factor  $M$  cannot be a prime number because if it was, the operation wouldn't be able to be divided into more than one stage. The more prime factors  $M$  contains, the more choices the designer has. Decimating for  $M=18$  can be done in different ways:

- one stage: 18
- two stages: 9 and 2, or 6 and 3
- three stages: 3, 3, and 2

**3.6.2 Why is using multiple stages interesting?** When decimating, when combining filtering and down-sampling, the computational and memory requirements of the filters can usually be reduced by using multiple stages.

**3.6.3 How can the optimum number of stages and the decimation factor of each stage be chosen?** The answer varies depending on several factors. Therefore, an evaluation of the resource requirements of each possibility must be done.

In spite of this, there are some rules of thumb which might help narrow down the choices:

- Use two or three stages.
- Decimate in the order from the largest to smallest factor. For instance, if  $M=30$  and we want to use three stages, decimate by 5, then by 3, then by 2.

## CHAPTER 4

## SAMPLING RECEIVERS

With the tools and theory explained in chapter 3 now it is time to see how all these parts are put together to build up a receiver. The first of the two different techniques that are analyzed within the chapter is the Sub-sampling receiver.

#### 4.1 Sub-sampling receiver [7]

An example of a Sub-sampling receiver, which is very well studied in paper of [7], is the first architecture to be analyzed. It is composed of a “Sample & Hold” and three identical “Decimation blocks”, each one down-sampling both the signal and the sampling frequency to half of the initial value. A block diagram of the circuit is shown in Figure 4.1.

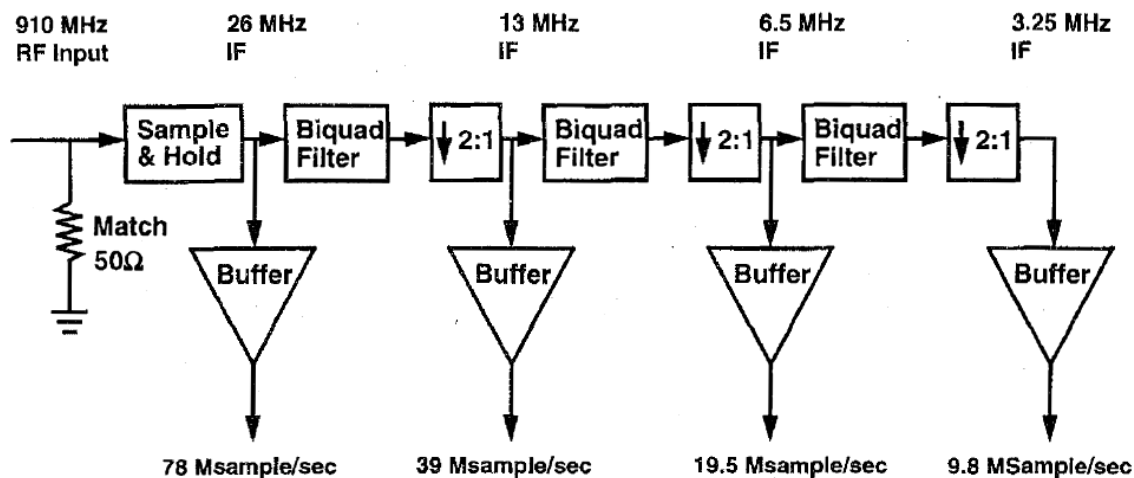


Figure 4.1. Block Diagram of the Discrete-time Signal Processing used in the Sub-sampling Receiver [7]

The input signal is located at 910 MHz, while the output of the last stage is at 3.25 MHz. The functionality of the whole architecture goes as follows: First, the spectrum is filtered by the analog devices placed before the Sample & hold (See Fig. 2.4). This operation and the resultant signal can be seen in Figures 4.2.a) and Figure 4.2.b). The signal obtained is cleaner since the out-of-band spectrum was attenuated. After filtering, the sub-sampling is performed at a much lower frequency than the sampling frequency. Thus, when sampled at 78 MS/s, many replicas of the initial signal appear at frequencies of:

$$f_i = nf_s \pm f_c$$

4.1

where  $f_i$  is the frequency of the signal replicas,  $f_s$  is the sampling frequency,  $f_c$  the carrier frequency and  $n$  is an integer. This effect is seen in Figure 4.2.c).

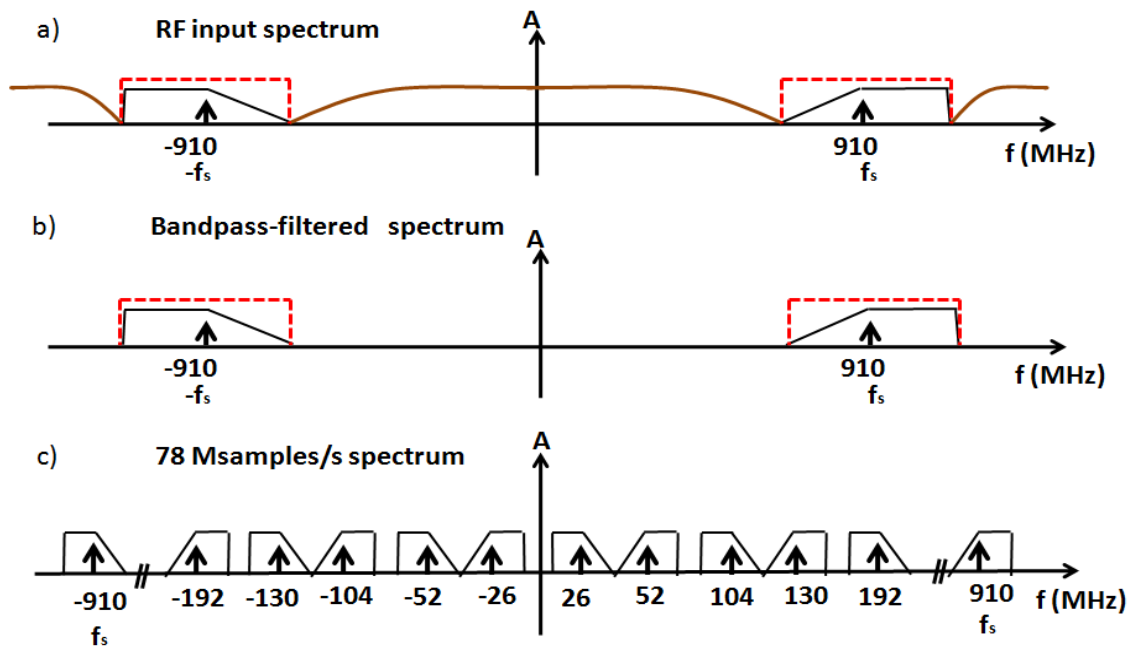


Figure 4.2. Effects of the sub-sampling operation in the frequency domain



Since  $f_c$  is not a multiple of  $f_s$ , direct sampling doesn't happen, but a down-conversion of the signal to a frequency much closer to baseband. In this case, where  $f_c=910$  MHz and  $f_s=78$  MS/s, the signal is moved to  $f_i=26$  MHz.

Figure 4.3 shows the signal's behavior as the signal is decimated (just the first decimation stage is showed). Here, we can see how the signal is down-converted to a half of the input frequency as it is decimated. The Anti-aliasing Filter (in red) filters out the aliasing-band so that once the signal is down-sampled no aliasing exists. Two more identical stages follow the one showed below, down-sampling the signal again until its spectrum is placed at 3.25 MHz and the sampling frequency is 9.8 MS/s.

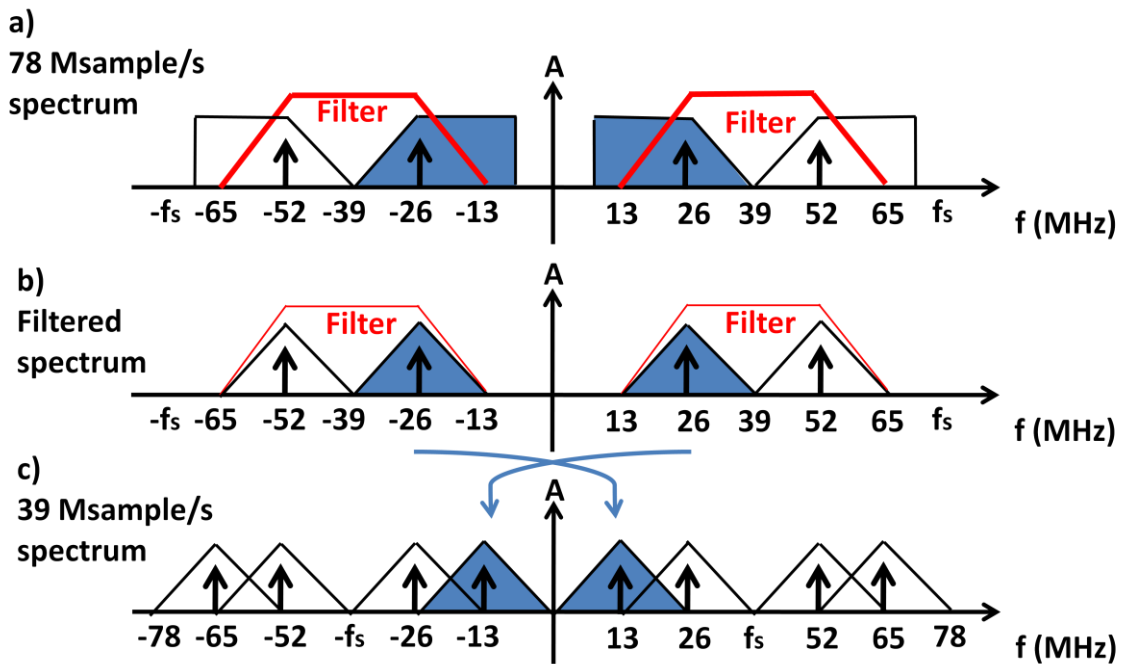


Figure 4.3. Effects of the decimation in the frequency domain

The Sample&Hold circuit can be seen at [7] and it is a fully differential switched-capacitor circuit. Its bandwidth is approximately 950 MHz. A biquadratic filter is used to implement all three 2-1 down-sampling stages. It has a single notch in the stopband that serves as Anti-aliasing filter as long as the channel bandwidth is narrow compared to its input sampling rate and to the notch width. The frequency response, hence, forms a notch at  $1/6$  of the sampling rate (where the signal is going to be placed) and has a gain of approximately 12.5 dB at  $1/3$  of the sampling rate (where the signal is initially). The Biquadratic Filter's frequency response is shown next (See Fig. 4.4):

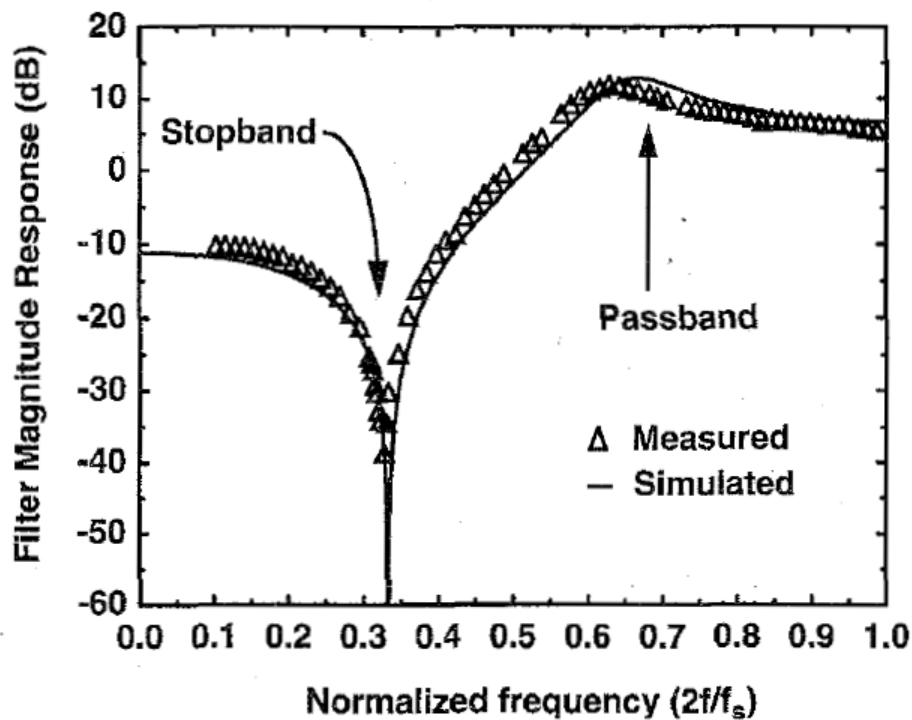


Figure 4.4. Biquadratic filter frequency response [7]

## 4.2 Direct-sampling receiver [10]

The second architecture under analysis, called “All-Digital receiver for Bluetooth Radio” uses direct conversion instead. Figure 4.5 presents the implementation of this receiver which is explained in [1]. The block diagram below includes both the phase and the quadrature paths, but for simplicity I am just going to focus in one of them since the structure is the same. It interesting to see how each path has also three decimation stages between the sampler and the ADC. However, in this case the three filters are different. The input signal, whose carrier is at 2.4 GHz, is sampled at 2.4 GS/s and hence, down-converted directly to baseband. Afterwards, the overall decimation factor applied to the sampling frequency is 64 ( $8 \cdot 4 \cdot 2$ ). Thus, the ADC works with a 37.5 MS/s sampled signal. Figure 4.6 shows how the direct sampling moves the bandpass signal into the baseband.

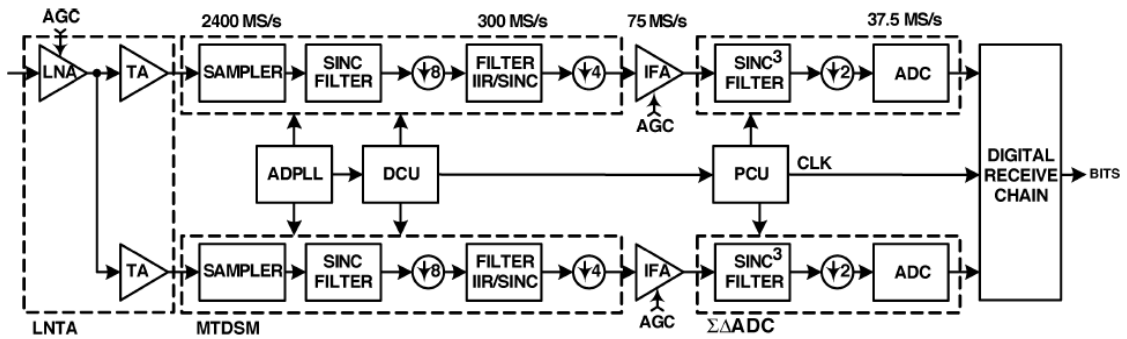


Figure 4.5. Block Diagram of the All-Digital receiver for Bluetooth Radio [10]

As seen in the Figure 4.7, the first Anti-aliasing filter, which is an 8-Sinc filter, eliminates the noise at multiples of  $f_s/8$ . Hence, when the signal is down-converted, its new location ( $f_s/8$ ) has no noise. This operation is done for every down-conversion at different decimation factors.

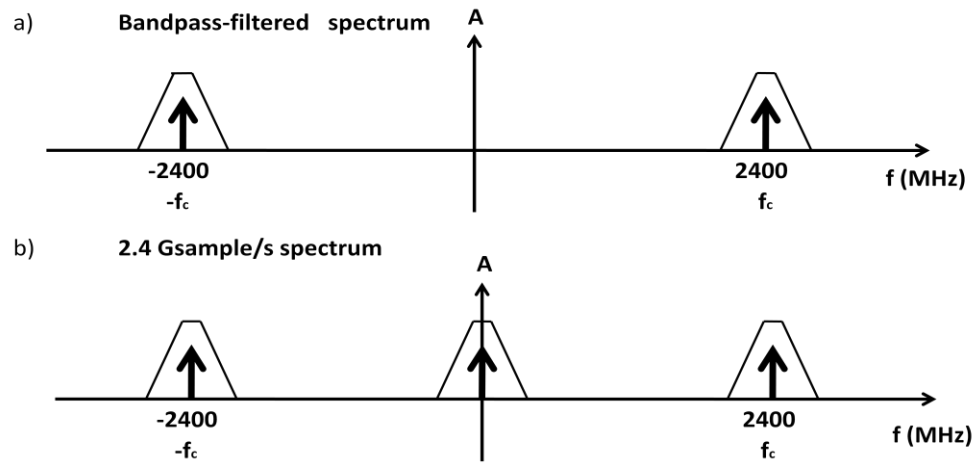


Figure 4.6. Effects of direct-sampling in the frequency domain

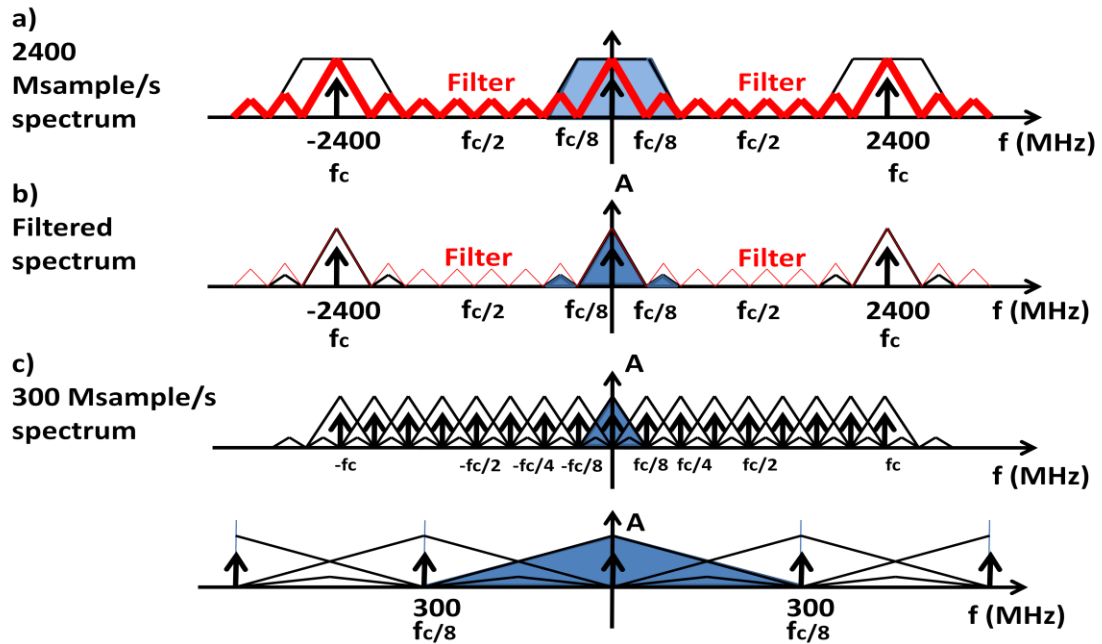


Figure 4.7. Effects of decimation in the frequency domain

The receiver is composed of two main blocks: The Multi Tap Direct Sampling Mixer (MTDSM)—which includes the Sampler, an 8-Sinc filter, an 8-down-converter, an IIR/4-Sinc filter and a 4-down-converter—and the Sigma-Delta ADC Converter—which includes a 2-Sinc<sup>3</sup> filter, a 2-down-converter and the ADC. These two blocks are explained next.

**4.2.1 Multi-Tap Direct Sampling Mixer (MTDSM).** The basis of the MTDSM's functionality is explained next (See Fig. 4.8): The current  $I_{rf}$  comes from the Low Noise Transconductance Amplifier (LNTA). MTDSM first samples the current that is going to be integrated for one of the two banks of 4 rotating capacitors. While one bank of capacitors is being discharged for the readout operation, the other is accumulating the N samples. This operation generates a frequency-sinc filter with decimation factor N. And since the readout is done as well every N samples the signal is at the same time down-sampled by N. In this case  $N=8$ .

Since doing the readout of the charge accumulated in the capacitors  $C_R$  (at 300 MHz) is difficult, the output charge readout time is extended by  $M=4$ . That is the reason why each bank has 4 capacitors. Now each capacitor is accumulating 8 samples and the readout operation of the whole bank is done after all 4 capacitors have been charged, what means after 32 samples. This operation creates another 4-sinc filter. Moreover, a history capacitor is introduced so that an IIR filter capability is introduced. Figure 4.8 shows the transistor level of the whole circuit:

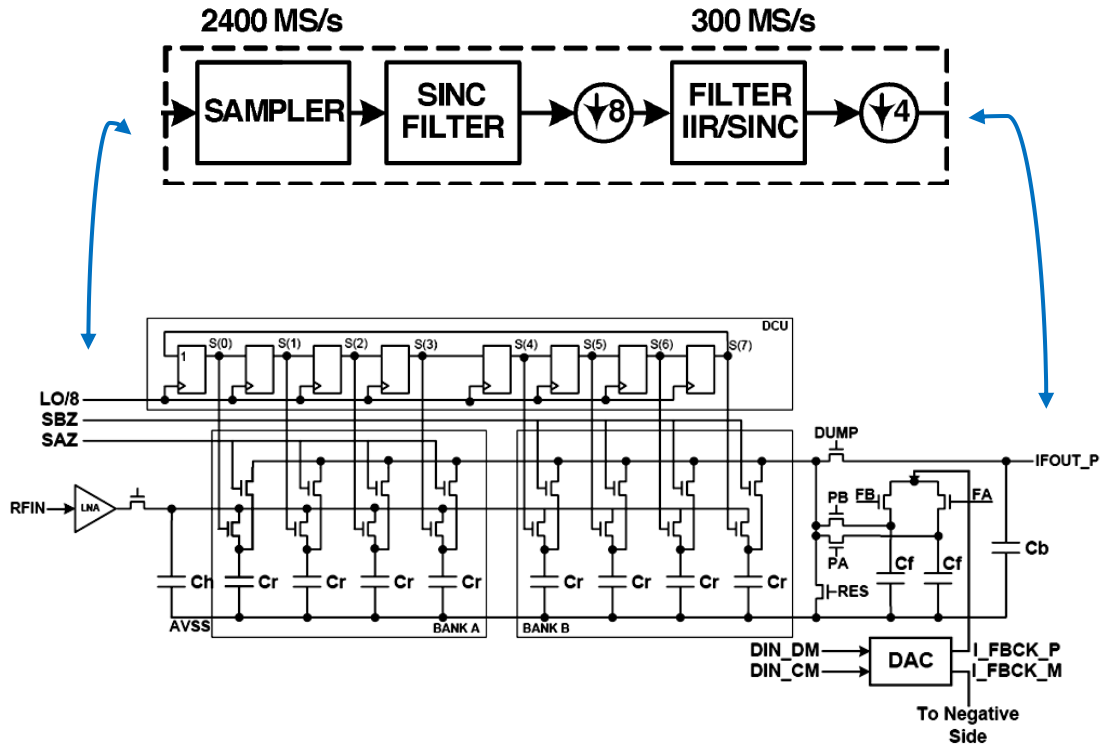


Figure 4.8. Complete transistor level of the MTDSM [10]

Now let's explain how the signal behaves step by step: (See Fig. 4.9-4.11). In the first period of time, the capacitor  $C_r$  of the first group of 4 capacitors (bank A) is getting charged. Meanwhile, the readout of the bank B is being done. During the second period of time the control signals don't change so nothing happens obviously a part from that the second capacitor is charged. In the third period the third capacitor is charged and bank B is reset. Therefore, there is no readout anymore and now the other  $C_f$  is precharged too ( $C_f$  are used to precharge the bank of capacitors before they receive the sampled current). During the fourth period the last capacitor is charged and the bank B, that had been readout and reset, is precharged by means of  $C_f$ . Operation of next 4 cycles is the same but interchanging bank B with bank A. Hence, while capacitors of bank B are being charged, bank A is going to be readout during periods 5 and 6, then reset and precharged.

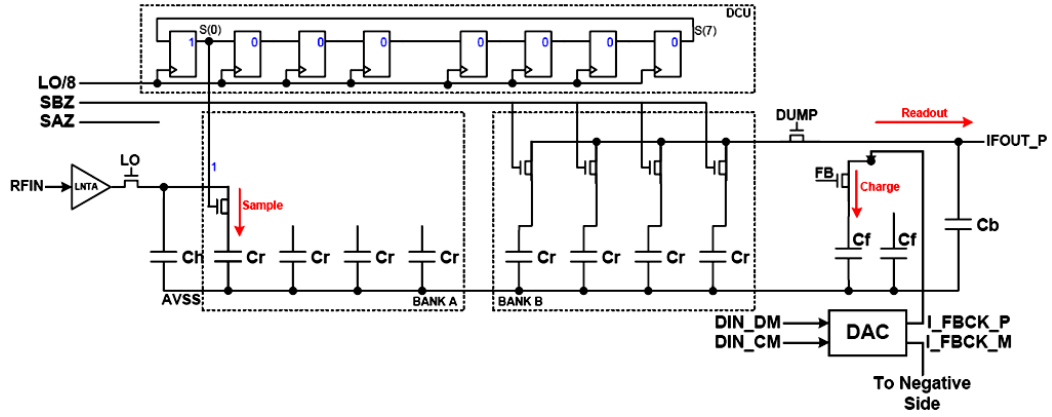


Figure 4.9. First step of the MTDSM operation [10]

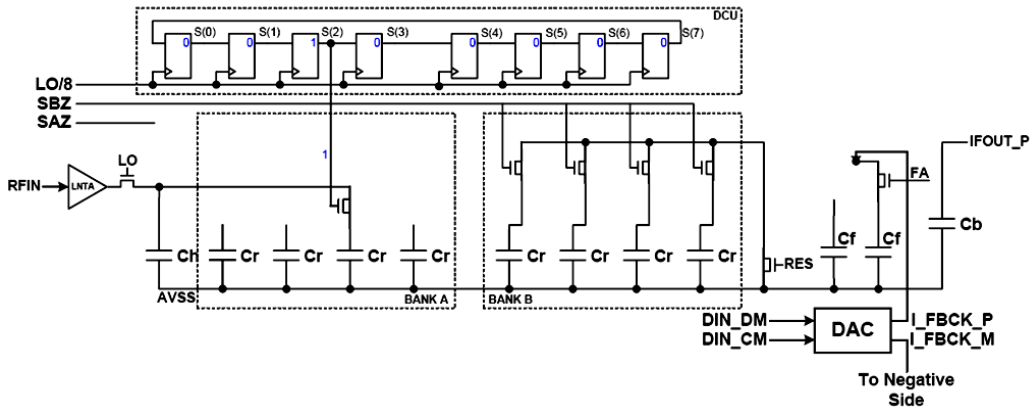


Figure 4.10. Third step of the MTDSM operation [10]

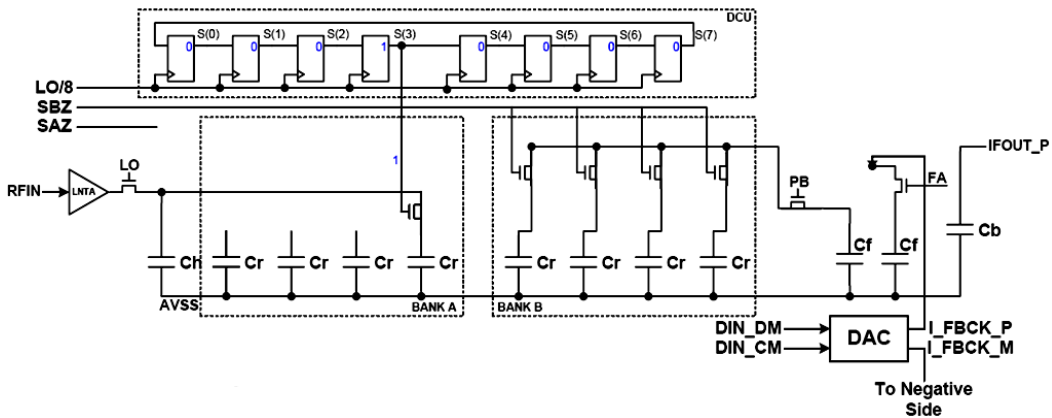


Figure 4.11. Fourth step of the MTDSM operation [10]

At the end of this cycle, the MTDSM frequency response is like the “First IIR Filter” signal shown in Figure 4.12:

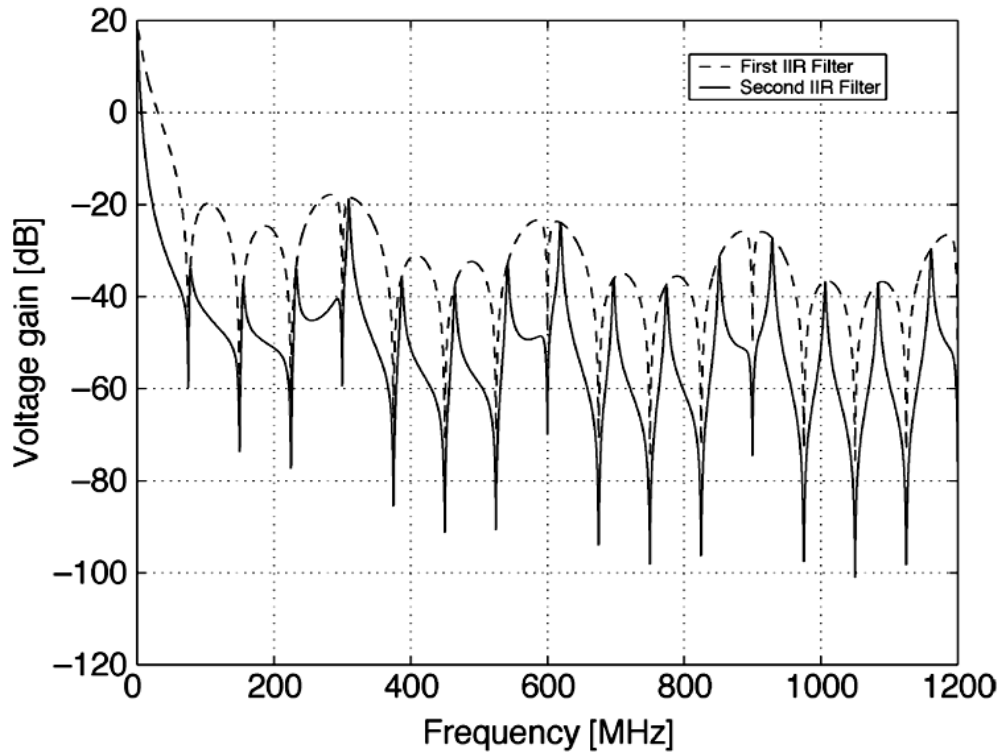


Figure 4.12. First and second IIR filters frequency response [10]

The first IIR filter is created as the combination of the two Sinc filters plus the “history capacitor”  $C_h$ , which produces the feedback to create an IIR filter. There is the 8-Sinc filter at 2.4 MS/s that produces the notches at 300 MS/s and there is the IIR/4-Sinc filter that produces the notches at 75 MS/s. The down-conversion places the output signal at 75 MHz, right where the first notch is located. Later, these filters are going to be simulated individually and combined.

The second IIR filter is produced because of the charge transference at the output during the reading operation. The effect of this second IIR is showed at Figure 4.12. The



voltage stored in the rotating capacitors cannot be readily presented to the MTDSM block output without an active buffer that isolates the high impedance of the mixer from the required low driving impedance of the output. The active element, which is an operational amplifier, does not actually take part of the IIR filtering process. It is merely used to sense voltage of the buffer feedback capacitor  $C_b$  and present it to the output with low driving impedance. The charge accumulated on the  $M$  rotating capacitors is being shared during the readout phase with the buffer feedback capacitor  $C_b$ . At the end of this phase, the  $M \cdot C_R$  capacitors are disconnected from the second IIR filter and their charge reset before they can be re-engaged in the MTDSM operation. This charge loss mechanism gives rise to IIR filtering.

**4.2.2 Sigma-Delta ADC with a built-in anti-aliasing filter [3].** This part includes the last stage of the discrete-time processing and the ADC. The block diagram of the Sigma-Delta ADC is showed in Figure 4.13:

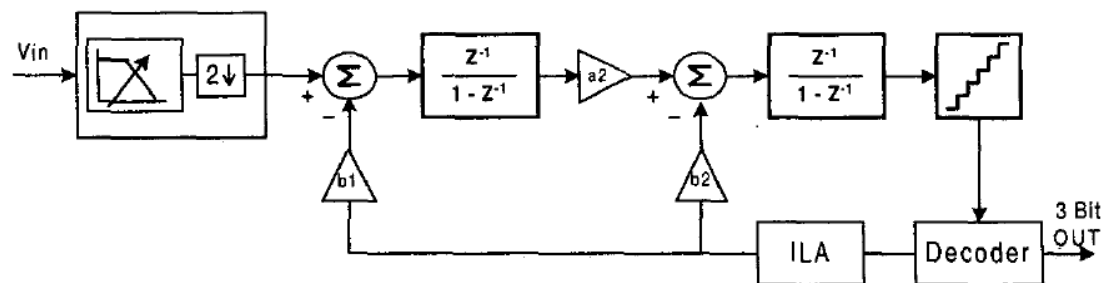


Figure 4.13. Sigma-Delta ADC with a built-in anti-aliasing filter: Block diagram [3]

The last decimation stage is already part of the Sigma-Delta ADC. Here, the anti-aliasing filter used is a  $2\text{-Sinc}^3$  filter. The circuit implementation and the control signals

can be seen at [7]. The third order charge domain FIR filter implementation is done using a switched capacitor sampling network. It includes a Gain Control by adding a high-gain (14dB) mode switched capacitor in parallel with each of the capacitors of the FIR filter. The signal at the input of the ADC is band limited by preceding circuits to 75MHz but the ADC works at half that frequency. Hence, the key role of the FIR filter is to provide enough noise suppression around  $F_s/2$ , that is 37.5 Mhz. The FIR filter equation is given by

$$y(n) = C_0x(n) + C_1x(n - 1) + C_2x(n - 2) + C_3x(n - 3),$$

4.2

where coefficients  $C_0$ ,  $C_1$ ,  $C_2$ , and  $C_3$  are 1, 3, 3, and 1 respectively. These values can be easily implemented as capacitor ratios.

### 4.3 Comparison between sub-sampling and direct-sampling receivers

To differentiate the basic elements of each of the two receivers described above we need to number each configuration: 1) Sub-sampling receiver and 2) Direct-sampling receiver. This way each point 5.x) will use 1) or 2) when referring to either the first or the second receiver.

#### 4.3.1 Anti-aliasing filter.

- 1) Biquad filter with a notch at  $f_s/6$  and maximum amplification at  $f_s/3$ .
- 2) Sinc (or  $\text{Sinc}^3$ ) filter with notches at multiples of the  $f_s/N$ , being  $N$  the decimation factor. It has a maximum (0 dB) at frequency  $f = 0$  Hz.

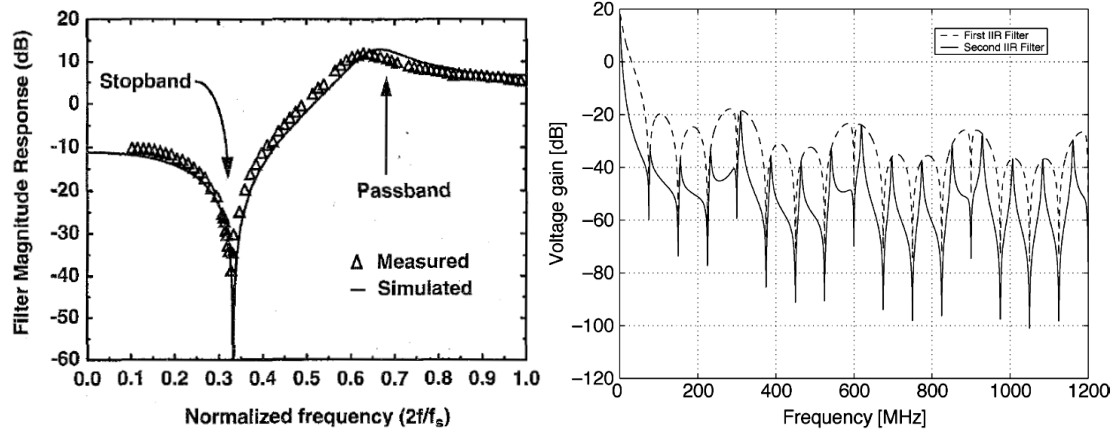


Figure 4.14. Left: Biquad filter frequency response [7]; Right: MTDSM combined frequency response of the first and second IIR filter [10]

#### 4.3.2 Down-sampling factor.

- 1) Always  $N=2$ , making the design easier since it uses the same filter every time.
- 2) It starts with  $N=8$  and it decreases by a factor of  $\frac{1}{2}$  each stage. The sequence is 8-4-2. Thus, there is a need of different circuits for every stage, at least the decimation factor. However, we have seen how the two first stages are integrated in the MTDSM and the last one is already part of the Sigma-Delta ADC.

#### 4.3.3 Frequencies used.

- 1)  $F_c = 910$  MHz,  $f_s = 78, 39, 19.5, 9.8$  MS/s,  $f_{if} = 26, 13, 6.5, 3.25$  MS/s
  - 2)  $F_c = 2.4$ GHz,  $f_s = 2400, 300, 74, 37.5$  MS/s,  $f_{if} = 2400, 300, 74, 37.5$  MS/s
- (DIRECT SAMPLING)

The main differences are actually the technology that the receiver has been built for, which determines the input signal frequency, and the technique used for the sampler – in this case sub-sampling or direct-sampling. The filter and decimation factor depend

on these two differences and are designed accordingly so that the signal and sampling frequency are within the specifications of the following stages, which are the ADC and the rest of the back-end stages. Hence, the first and basic problem is to choose what is better regarding the signal coming from the antenna. The following point explains the advantages of direct-converting.

**4.3.4 General Advantages of Direct Conversion.** Once the chip and system design have been determined in order to deal with the design issues of direct-conversion radios, there are advantages beyond the simplicity of the circuit, and one of them is the cost. Most of the added complexity to deal with direct-conversion limitations is inside the DSP chip, which only needs to be done once for millions of handsets, and will be fabricated on the low-cost silicon. Some other notable advantages include the following: There is no frequency limits since it can operate on any frequency; The front-end filter can even be eliminated if necessary; Swept receivers, such as those in spectrum and network analyzers can be implemented with this kind of techniques; It has minimal spurious response (caused by unwanted mixing products, nonlinearities and imperfect isolation); In a direct-conversion radio, the only significant spurious are at the harmonics belonging to the local oscillator (LO), which are far removed from the operating frequency; And high linearity, due to the short path the signal goes through.

#### **4.4 Simulink Model**

To simulate the ideal function of the circuit, a Simulink (MATLAB®) model has been built. Before explaining the complete model in detail, each filter is presented:

**4.4.1 Sinc filter (N=8).**

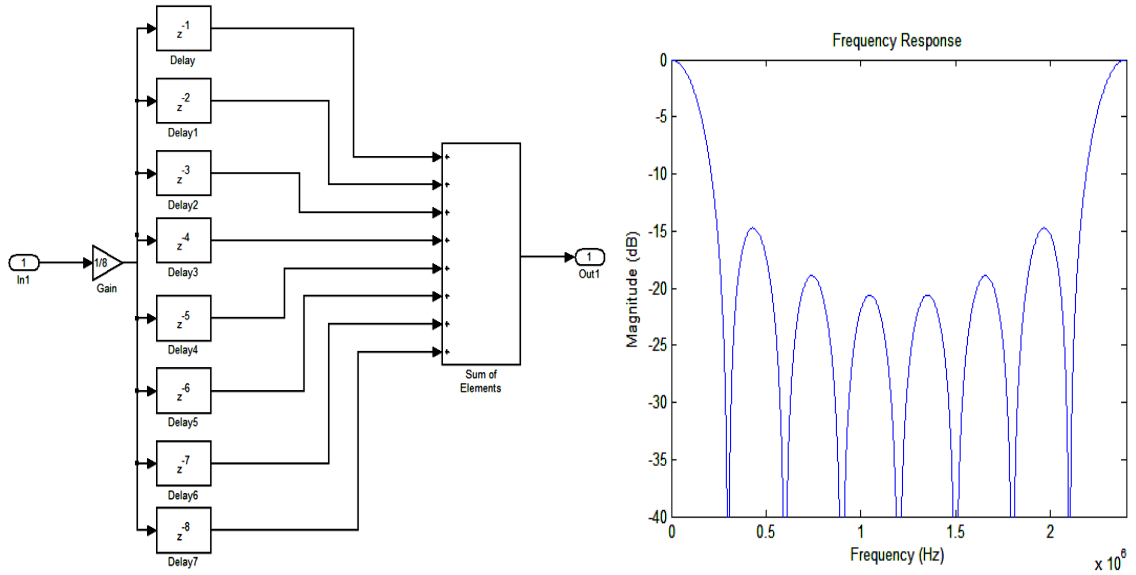


Figure 4.15. Left: 8-Sinc filter implementation; Right: Frequency Response

**4.4.2 Sinc filter (N=4).**

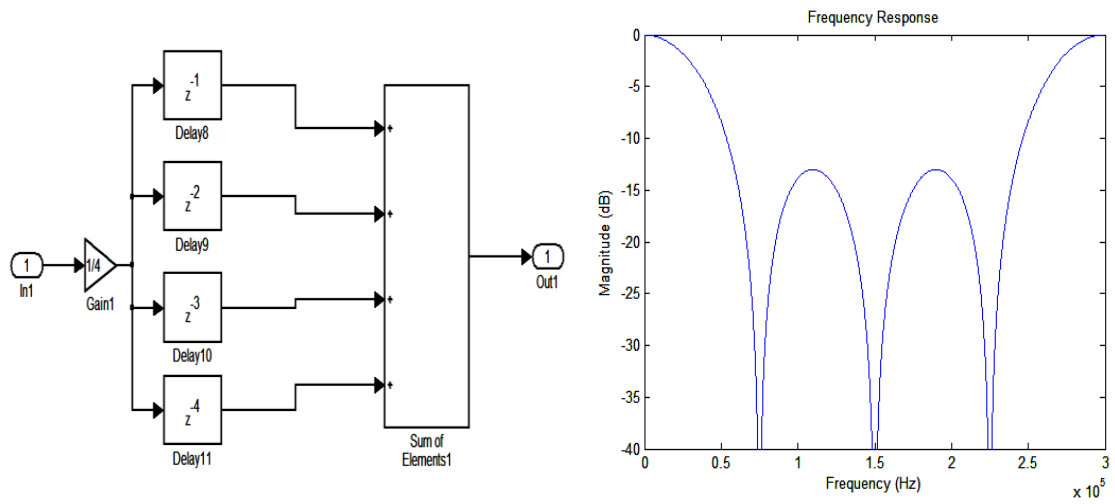


Figure 4.16. Left: 4-Sinc filter implementation; Right: Frequency Response

#### 4.4.3 Sinc<sup>3</sup> filter (N=2).

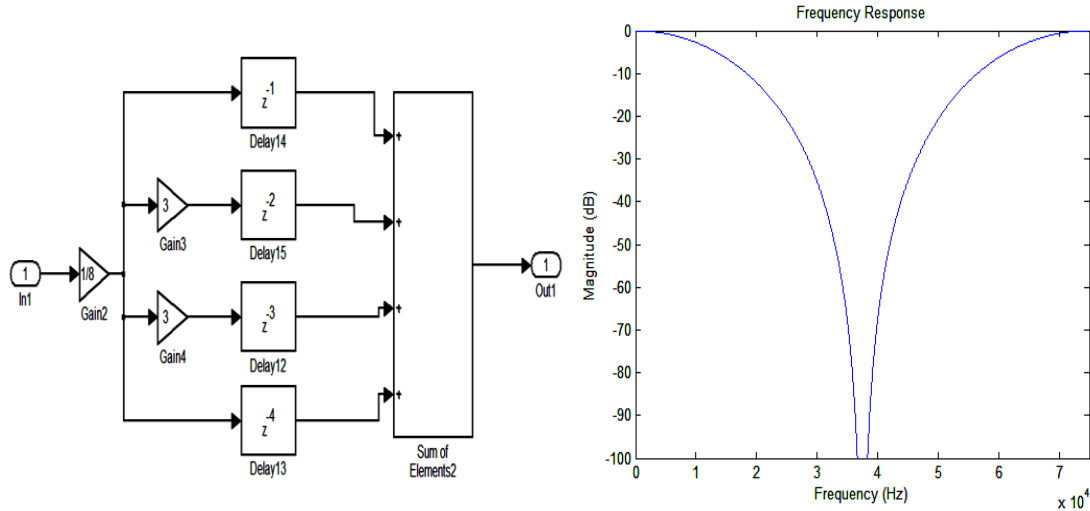


Figure 4.17. Left: 2-Sinc<sup>3</sup> filter implementation; Right: Frequency Response

These implementations come from the formula of eq. 4.3.

$$y(n) = \left\{ \frac{1}{N} \sum_{n=1}^{N-1} x(n-i) \right\}^M$$

4.3

Note: To see the code used in Matlab for plotting the signals obtained in the Simulink Model go to Annex A.

**4.4.4 MTDSM.** The frequency response of the whole Multi-Tap Direct Sampling Mixer is the combined response of the 8-Sinc filter together with the IIR/4-Sinc filter, at the same time that the signal is down-converted with N=32. To find the MTDSM response I built another Simulink file (See Fig. 4.18).

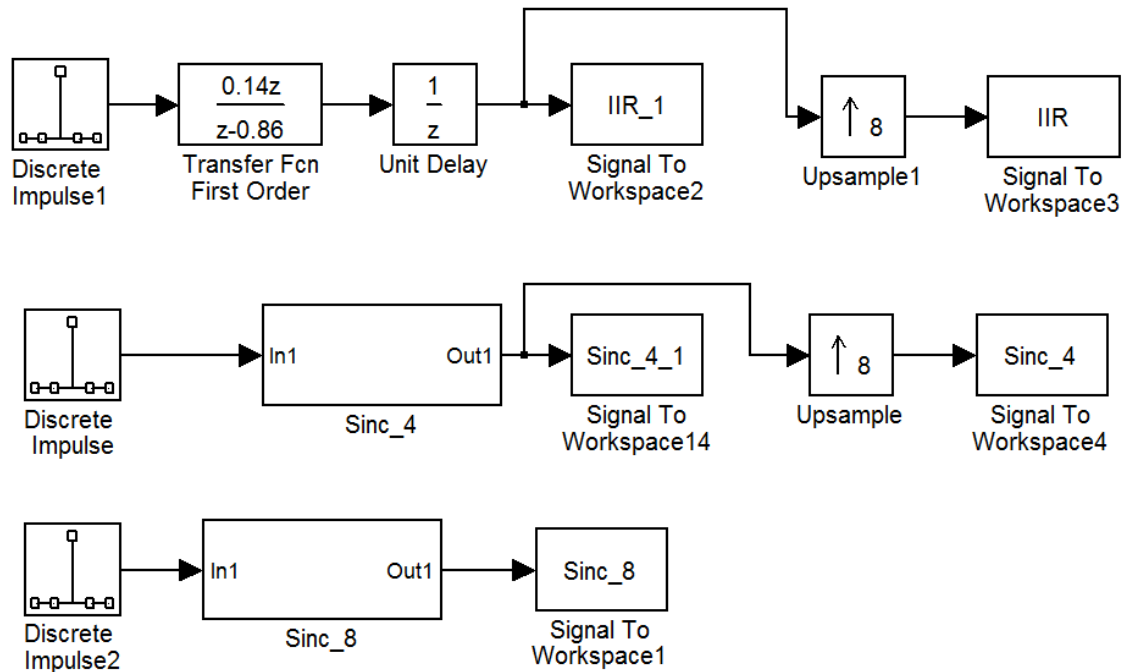


Figure 4.18. MTDSM Simulink block diagram

This model is aimed at finding the Anti-aliasing filter's response. In figure 4.18 there are three different block diagrams. The diagram on top acts as an IIR filter and its values have been found experimentally to have the same curve that the one showed the paper [10]. The second and third diagrams simulate the 4-Sinc filter and the 8-Sinc filter. Since the sampling frequencies for each filter are not the same, an up-sampling stage with decimation factor 8 is used to allow plotting all the signals together within an equal range of frequencies.

**4.4.5 Receiver.** Gathering all the elements explained before, a model of the whole system can be built (See Fig. 4.19). The following points explain the model in detail.

- 1) A “Bernoulli Binary Generator” which generates 1’s and 0’s with equal probability has been used. This binary signal is modulated by a “BPSK Modulator Baseband” block and multiplied by a sinusoidal carrier of 2.4 GHz.
- 2) This signal goes through an “AWGN Channel” which introduces white Gaussian noise.
- 3) Then the “Real” part is selected (BPSK signals are real, but the noise introduced for the AWGN Channel in the model could add imaginary values).
- 4) Instead of a “Sample&Hold” a down-converter with  $N=10$  is used since the signal in Matlab® is always discrete and the rate that the signal was modulated at was 10 times the sampling frequency of the receiver. Another component could have been used but a down-sampler was the one that modified the spectrum the least and its function was exactly the one required.
- 5) Once the signal is sampled at the sampling frequency of 2.4 GS/s it goes through the whole block diagram studied before: an 8-Sinc filter, an 8-down-sampler, an IIR filter (whose parameters have been found analytically to have a close response to the one in the papers), a 4-Sinc filter, an 4-down-sampler, another IIR filter (which has been designed with the same parameters as the first one), a 2-Sinc<sup>3</sup> filter, and a 2-down-sampler.



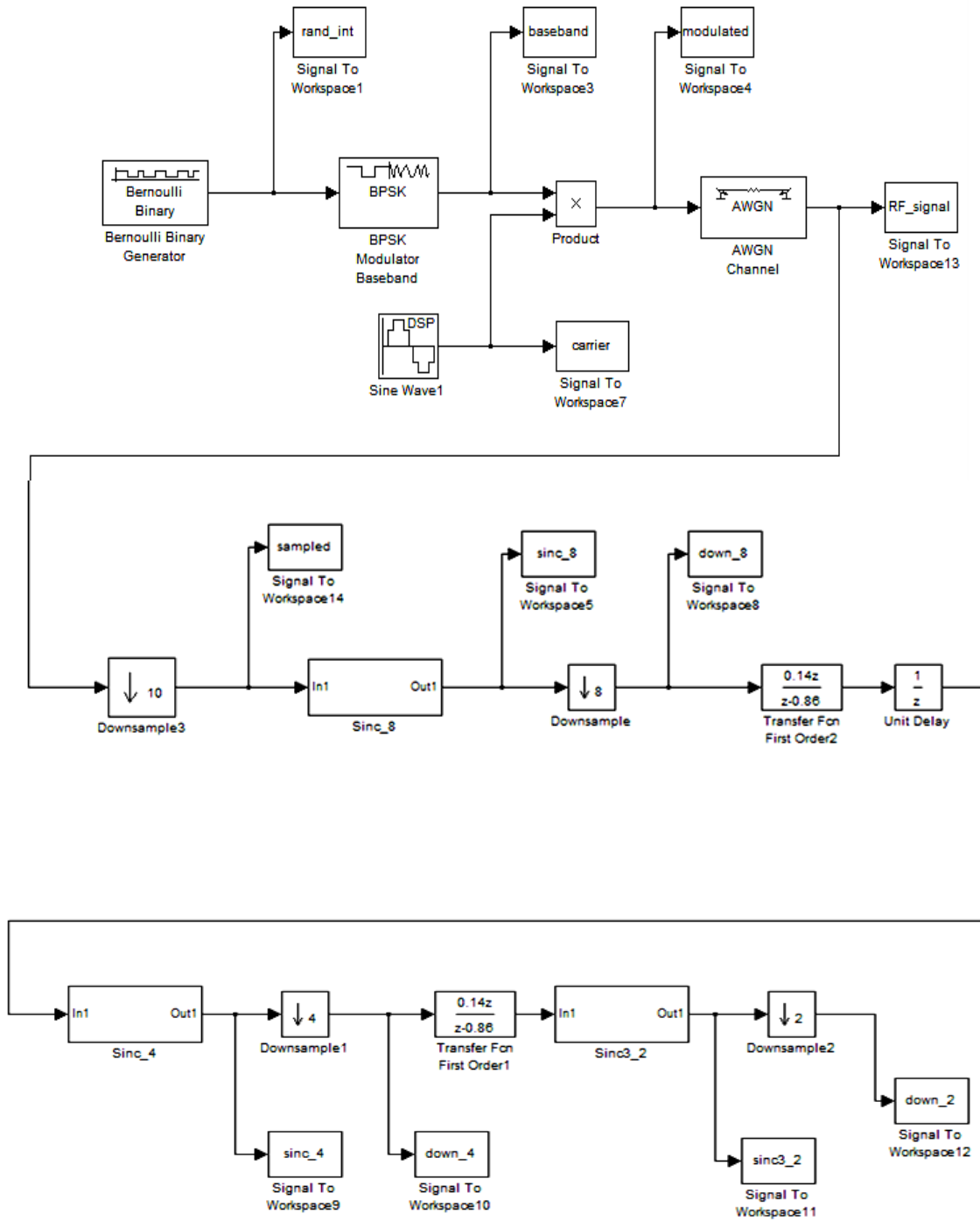


Figure 4.19. Receiver Block Diagram (Simulink)

### 4.5 Results

The MTDSM and the whole receiver have been simulated in this section. For the MTDSM, all the signals IIR, Sinc\_4 and Sinc\_8 obtained from the model in Figure 4.18 have been plotted together to see what notches each filter introduces. These waveforms are shown in Figure 4.20.

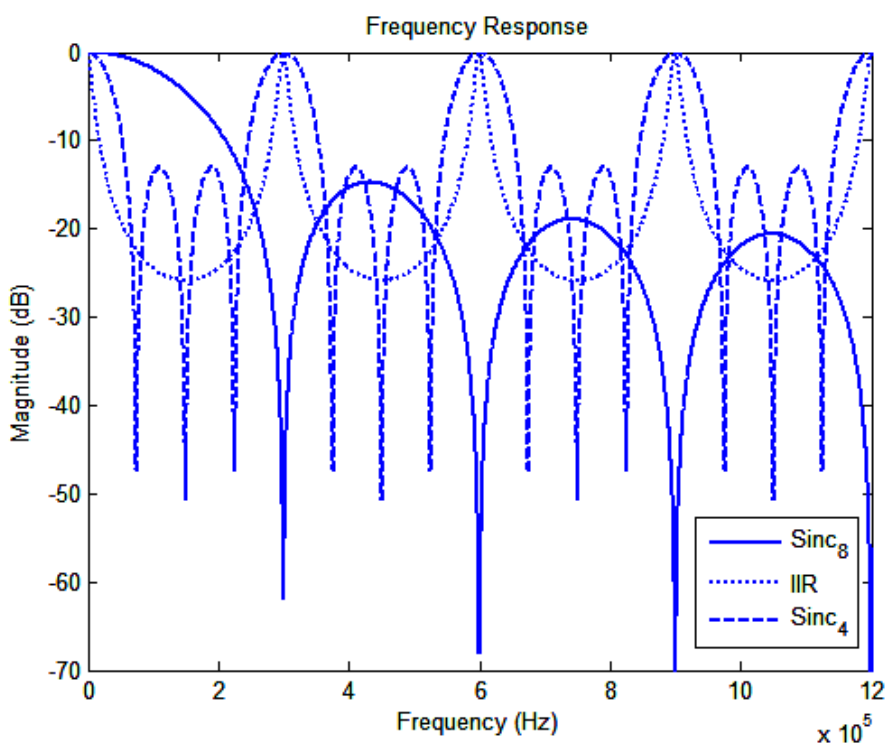


Figure 4.20. Frequency response of each filter included within the MTDSM

Combining these three signals—multiplying the frequency response—we obtain its complete frequency response that can be seen in Figure 4.21. Its form matches perfectly with the simulation done in [10] (See Fig. 4.12).

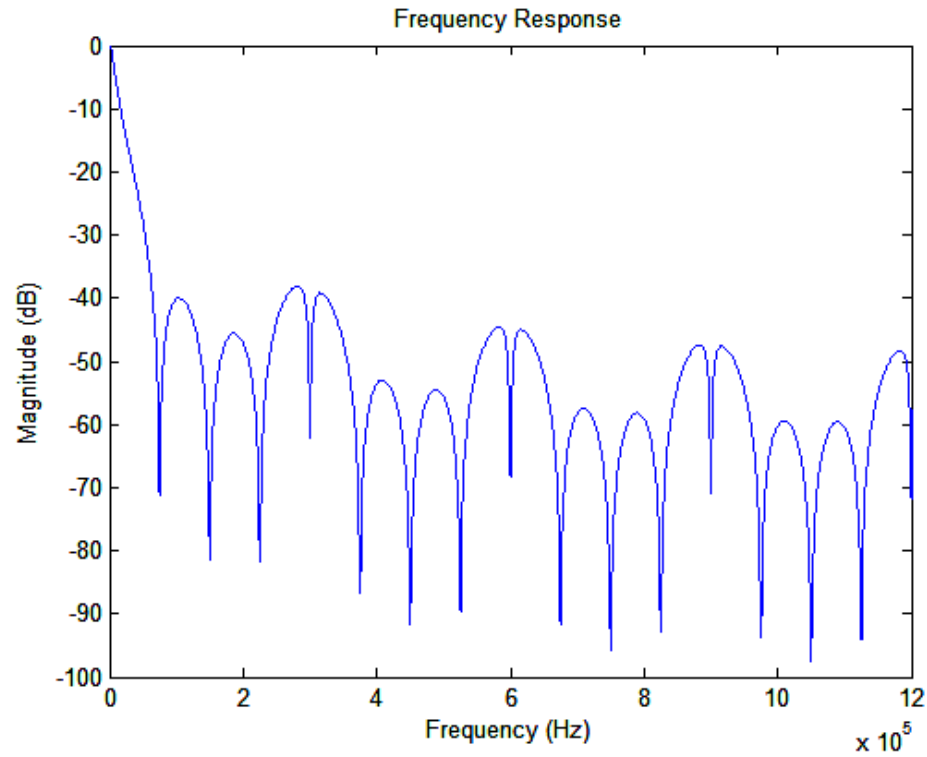


Figure 4.21. MTDSM frequency response

Note: To see the code used in Matlab for plotting the signals obtained in the Simulink Model go to Annex A.

For the second model, done for simulating and observing the behavior of the complete receiver, the results obtained display the signal at each point in the frequency and time domain. The important point here is not to see the transfer function but to see how all the decimation stages join efforts to down-convert and clean the signal so that the initial baseband signal is obtained at the end. Figures 4.22-4.23 show the results.

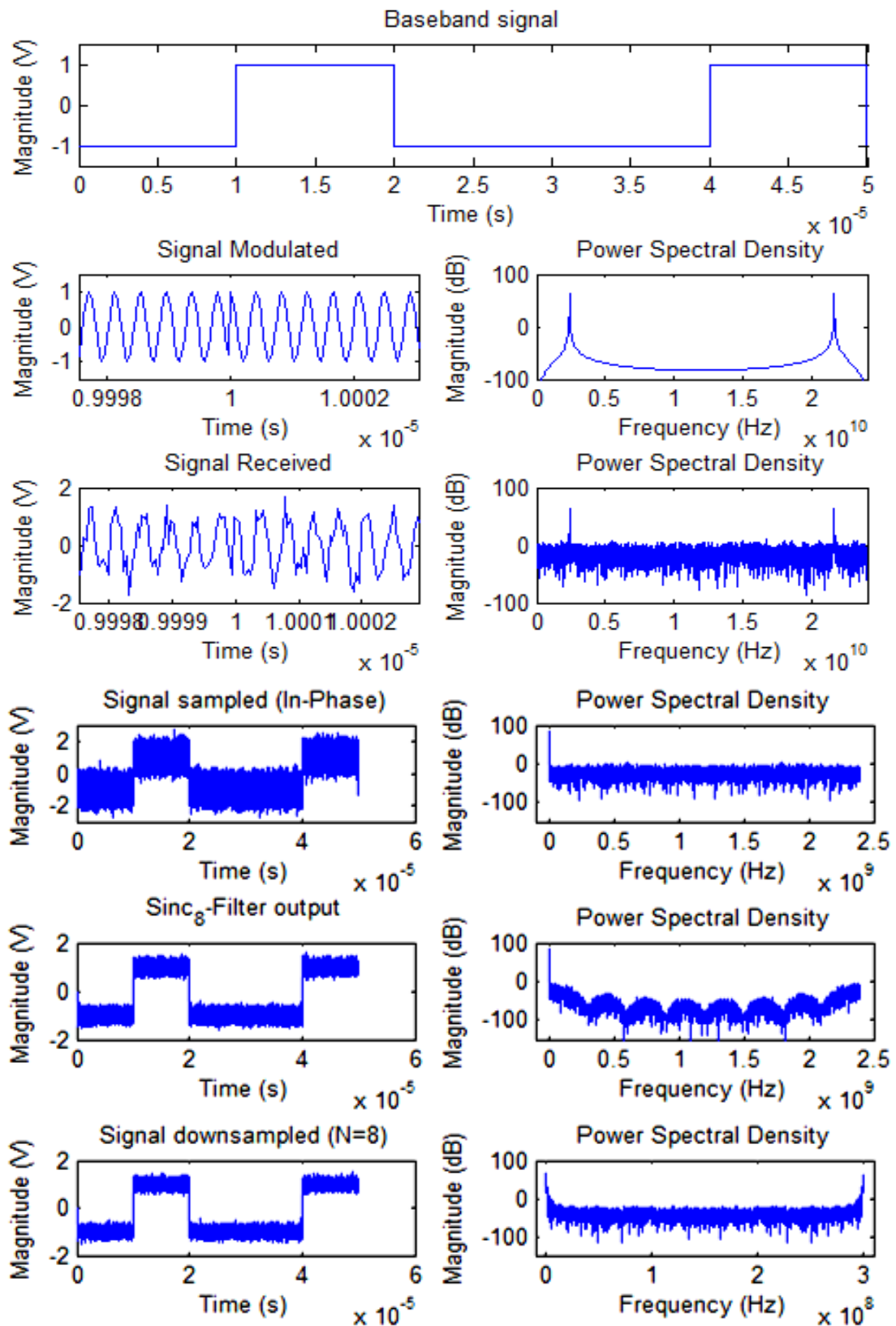


Figure 4.22. Simulation results\_1

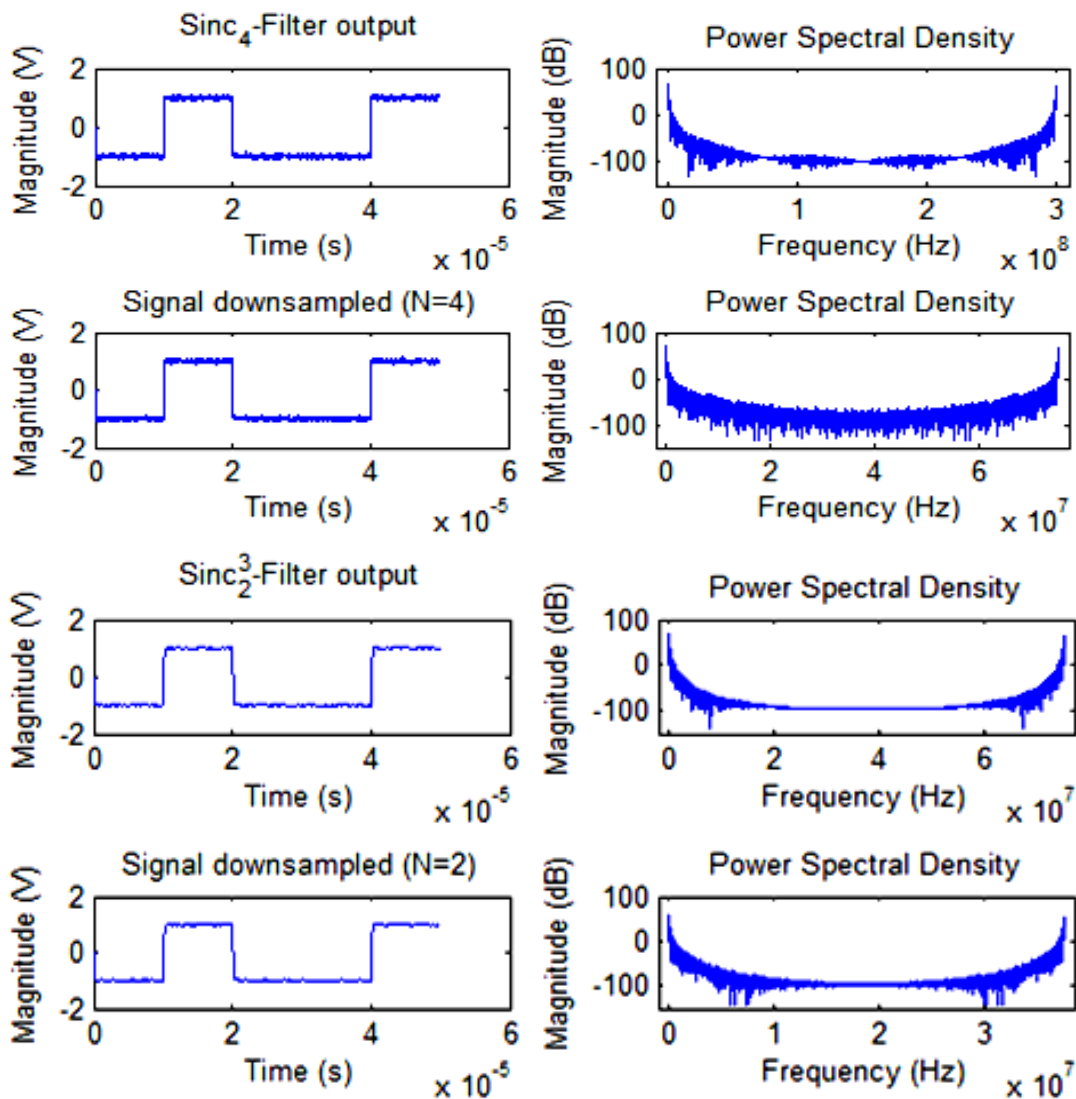


Figure 4.23. Simulation results\_2

Note: To see the code used in Matlab for plotting the signals obtained in the Simulink Model go to Annex A.

From the results one can see all the steps the signal goes through until it reaches the ADC. Each row shows the time and frequency domain of the signal within the corresponding step. There are a total of ten steps showed including the baseband, modulated, received, sampled, filters' outputs, and down-converted signals. The left hand side of the two columns shows how the noise is being eliminated as the signal is filtered

up to three times, and how the waveform happens to be neater. The right hand side, instead, shows the effect of the location of the notches. For example, it is easy to see the 7 notches introduced by the 8-Sinc filter, or how the noise is almost inexistent at multiples of 75 MHz after the 4-Sinc filter and at half of the sampling frequency after the 2-Sinc<sup>3</sup> filter. The overall effect is actually the one wanted: a lower sampling frequency with suppression of most of the noise. I must clarify that the signals that appear in the “Signal modulated” and “Signal received” Figures should be time-continuous and hence just one signal located at 2.4 GHz should appear. Again, this happens because Matlab® works with discrete-time signals and instead of having a time-continuous signal, it is actually a discrete-time signal sampled at 24 GHz.

#### **4.6 Conclusions**

The model created in Simulink uses a basic BPSK modulation to show graphically its correct behavior. However, the receiver is aimed at Bluetooth, which implies using GFSK instead. If GFSK, with all the new innovations for Bluetooth, had been used, observing the signal at each step in the time domain would have been much harder, and this was not actually the main purpose. Moreover, any modulation could have been used. Therefore, this file can be used in the future to prove the validity of this architecture for other modulations, to measure some parameters’ behavior—like for example to measure the evolution of the signal to noise ratio, taking into account that the receiver is ideal— or to set up the parameters for the new configurations.

## CHAPTER 5

## A RECONFIGURABLE ANALOG BASEBAND FILTER

## 5.1 Introduction

This chapter narrows down the analysis made in the previous sections—where complete receivers were analyzed—to a unique component, a configurable filter useful for Software-Defined Radio. This part is focused on the filter explained in the paper “A Widely-Tunable, Reconfigurable CMOS Analog Baseband IC for Software-Defined Radio” [2]. It explains and simulates the components of this reconfigurable filter analyzing their functionality. Figure 5.1 shows the block diagram of the whole device.

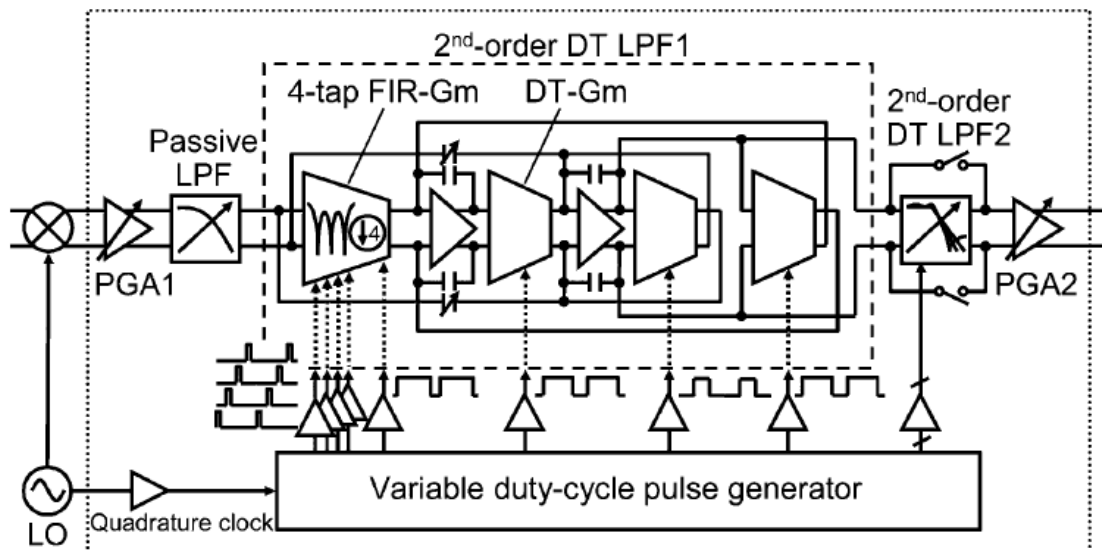


Figure 5.1. Block diagram of the analog reconfigurable filter [2]

The filter itself is composed mainly by two second-order Discrete-time LPF's (DT-LPF), a passive LPF before them and two programmable gain amplifiers (PGA), one

at the beginning to tune the coarse gain (PGA1) and another at the end of the whole circuit tuning the fine grain (PGA2). The remaining control block, and one of the most important parts, is the “Variable Duty-cycle Pulse Generator”. This block generates the control signals (or clocks) that rule every single synchronous part in the circuit. The gain of the configurable blocks, like the Transconductance stages, depends on the duty-cycle of these clocks. Hence, the device can be easily programmed. Figure 5.2 presents how different duty-cycle clock signals are generated.

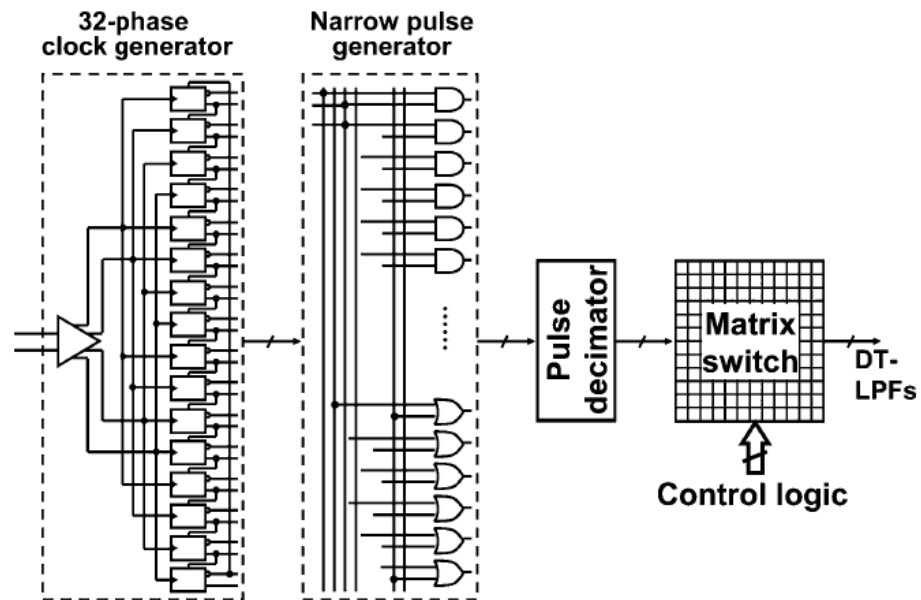


Figure 5.2. Variable duty-cycle pulse generator [2]

The Variable Duty-cycle Pulse Generator consists of a 32-phase clock generator, a narrow pulse generator, a pulse decimator and a matrix switch controlled by logic. The 32-phase clock generator creates 32 outputs each one shifted  $1/32$  of the period from each other. That was achieved by using 32 D-type flip-flops concatenated (each output is the



input of the next flip-flop), and each flip-flop being driven for a  $90^\circ$ -shifted clock from the previous one (using a 2GHz 4-phase input clock). Therefore, 32 phase-shifted clocks, of duty-cycle equal to 50% and frequency 250 MHz (2 GHz divided by 16), are obtained. The narrow pulse generator provides duty-cycles from  $1/32$  to  $31/32$  by using AND/OR operations of the output signals from the previous stage. If a very narrow bandwidth filter is required, the pulse decimator generates very low duty-cycle pulses down to  $1/4096$ . The decimation is done by an AND operation with a narrow-pulse and a divided clock of the narrow-pulse itself. At the end some logic controls the matrix switches so that the clock wanted is chosen.

The first stage of the first DT- LPF is a decimation stage—4-tap FIR-Gm filter—with factor  $N=4$ . Figure 5.3 shows its schematic.

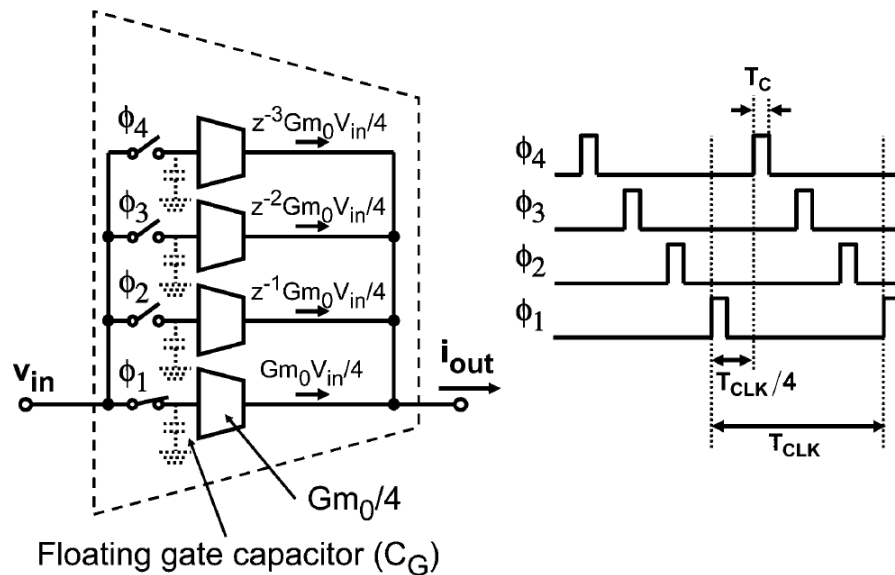


Figure 5.3. 4-tap FIR-Gm filter [2]

The four switches sample the input signal at 250 MHz, each clock with a phase shift of  $\pi/2$ —which is equivalent to a sampling frequency of 1GHz and introducing some delays—and the Transconductance stages that come afterwards acts as a flip-flop amplifying the signal by  $Gm_0/4$ . The sum of the current of all four wires yields the transfer function seen in Equation 5.1.

$$Gm_{FIR4}(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{4} Gm_{eff}$$

5.1

The reason of this first stage is to filter out the frequencies at  $f_{clk}$ ,  $2 f_{clk}$  and  $3f_{clk}$ . Together with the previous passive LPF, which introduces high attenuation at  $4f_{clk}$ , they form the Anti-aliasing filter so that when the signal is finally sampled at 250 MHz, all the frequencies that would move to baseband have already been filtered out. To achieve sufficient attenuation at the notches,  $T_c$  should be short compared to  $T_{clk}/4$ -  $T_c$  (where  $T_{clk}=1/250$  MHz).

The transconductance stages found four times in both Discrete-Time LPFs go always followed by an integrator, so that the former controls the gain while the latter introduces a pole. The schematic of a duty-cycle controlled Transconductor is showed in Figure 5.4. There are also plotted the expected control, input and output signals. This discrete-time transconductor consists of one inverter followed by two switches that sample the current. The duty-cycle of the control signals will fix the total current transferred and hence, the gain of this block. The combination of a couple of Gm-C (transconductor + integrator) stages, plus two feedback loops form a 2<sup>nd</sup>-order DT-LPF. Figure 5.5 presents the DT-LPF.

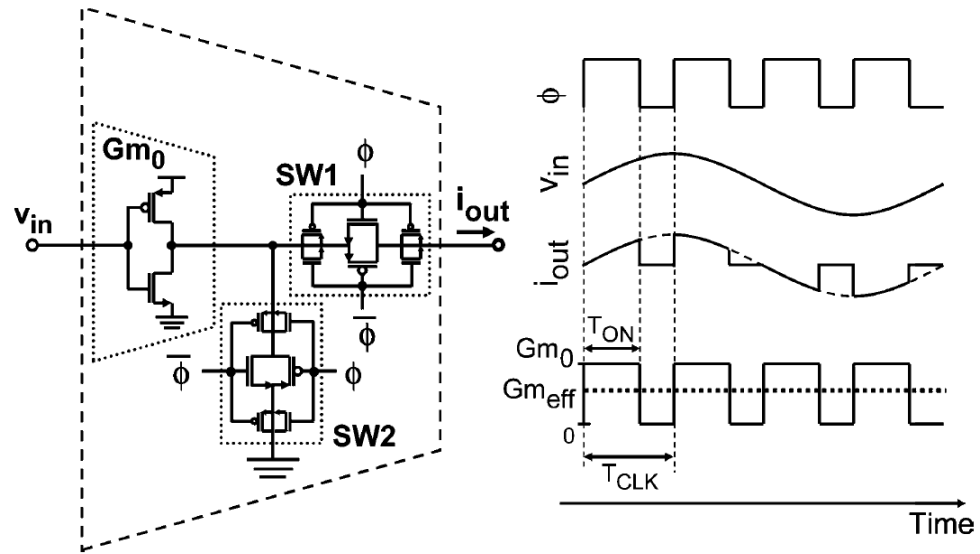


Figure 5.4. Duty-cycle controlled Discrete-time Transconductor [2]

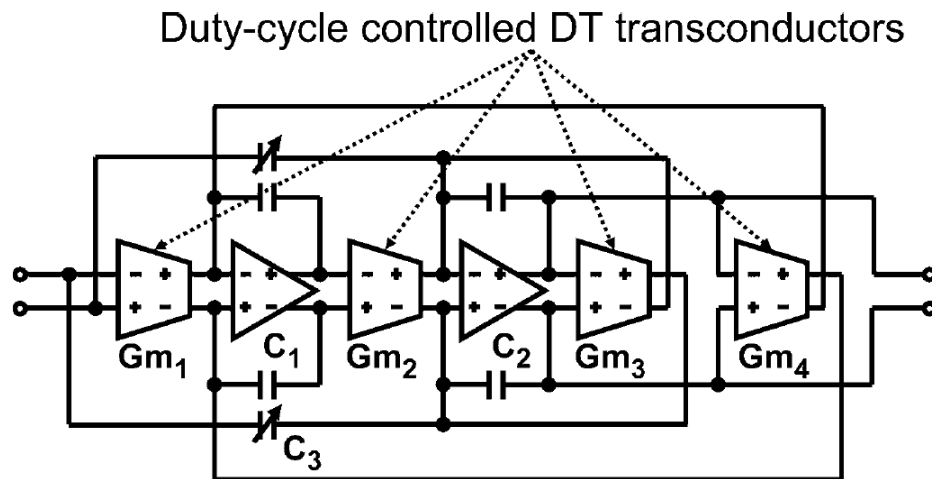


Figure 5.5. Duty-cycle controlled Discrete-time Low-pass filter [2]

Both DT-LPFs have the same architecture, but as said before, the first one has a 4-tap FIR filter instead of the first  $Gm$  block in order to achieve proper anti-aliasing

filtering. Figure 5.6 shows the block diagram of this second-order filter to easily understand the structure that it has.

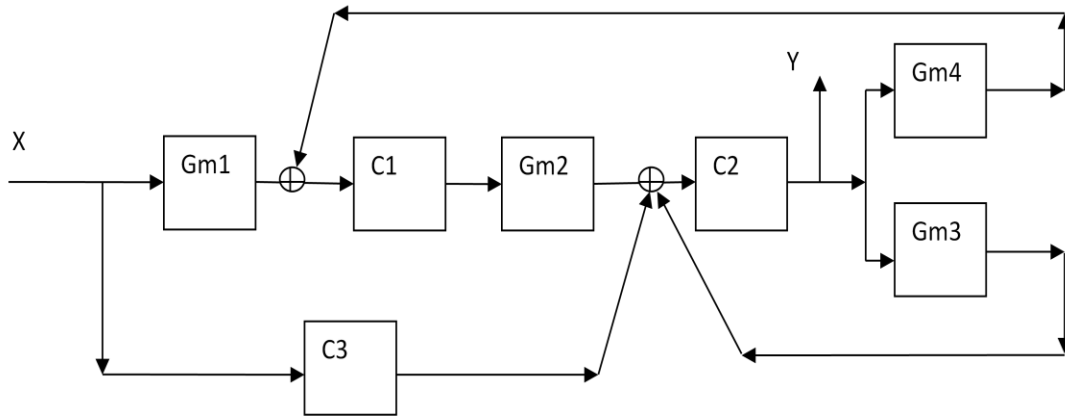


Figure 5.6. Functional Block diagram of the DT-LPF

## 5.2 Understanding the filter

In order to understand the whole block diagram it is necessary to identify what each block's transfer function is. For doing this I used the block diagram of Figure 5.7 with generic functions  $H_x(s)$ , developed the equations and found the transfer function of the diagram. Then, by comparing my result with the transfer function given in [10], I found the transfer function of each block. This transfer function depends on all the transconductances and capacitors (All the equations corresponding to these calculations are shown as Equation 5.2).

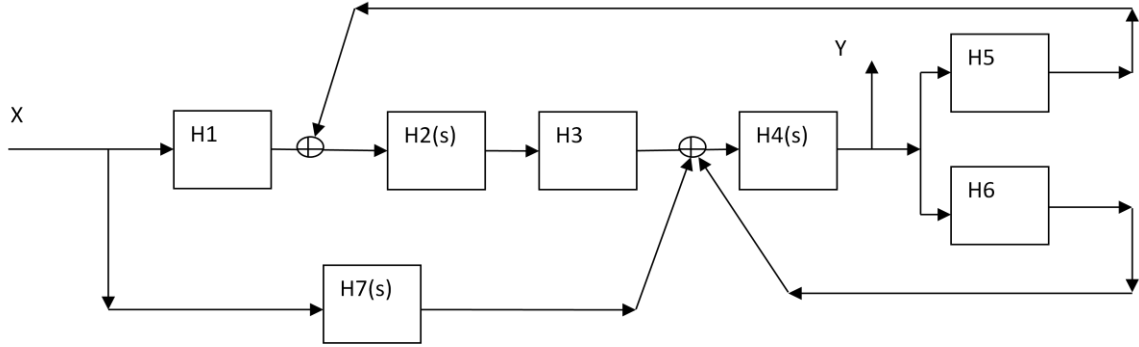


Figure 5.7. Generic diagram used for finding partial transfer functions

The block diagram of Figure 5.7 is mathematically analyzed and solved below:

$$y(s) = H4(s)(H6 * y + H7(s) * x(s) + H3 * H2(s)(y(s) * H(5) + H1 * x))$$

$$y(s) \left( \frac{1}{H4(s)} - H6 - H3H2(s)H(5) \right) = x(s)(H7(s) + H3H2(s)H1)$$

It is already known that the transconductance stage's transfer function is actually a constant, so the following substitutions can be done.

$$\mathbf{H1 = Gm1, H3 = Gm2, H6 = Gm3, H5 = Gm4}$$

$$y(s) \left( \frac{1}{H4(s)} - Gm3 - Gm2H2(s)Gm4 \right) = x(s)(H7(s) + Gm2H2(s)Gm1)$$

$$T(s) = \frac{y(s)}{x(s)} = \frac{H7(s) + Gm2H2(s)Gm1}{\frac{1}{H4(s)} - Gm3 - Gm2H2(s)Gm4}$$

$$T(s) = \frac{\frac{C3}{C2}s^2 + \frac{Gm2Gm1}{C2C1}}{s^2 + \frac{Gm3}{C1}s + \frac{Gm2Gm4}{C2C1}}$$

Now, knowing that the remaining functions are integrators and comparing the solution with our last step we can say:

$$H2 = \frac{1}{C1s}, H4 = \frac{1}{C2s}$$

$$T(s) = \frac{H7(s) + \frac{Gm1Gm2}{C1s}}{C2s - Gm3 - Gm2 \frac{1}{C1s} Gm4} = \frac{\frac{H7(s)}{C2} + \frac{Gm1Gm2}{C2C1s}}{s - \frac{Gm3}{C2} - \frac{Gm2Gm4}{C2C1s}}$$

$$= \frac{\frac{H7(s)}{C2}s + \frac{Gm1Gm2}{C2C1}}{s^2 - \frac{Gm3}{C2}s - \frac{Gm2Gm4}{C2C1}}$$

The only function left must be equal to:

$$H7 = C3s$$

$$T(s) = \frac{\frac{C3}{C2}s^2 + \frac{Gm1Gm2}{C2C1}}{s^2 - \frac{Gm3}{C2}s - \frac{Gm2Gm4}{C2C1}}$$

In order to have the same sign  $H4 = -\frac{1}{C2s}$  then,

$$T(s) = -\frac{\frac{C3}{C2}s^2 + \frac{Gm1Gm2}{C2C1}}{s^2 + \frac{Gm3}{C2}s + \frac{Gm2Gm4}{C2C1}}$$

5.2

Now, I already now which are the corresponding functions of each block. Therefore, I can set up the filters so that their response is the one desired. The following points are going to explain part of the theory behind the three configurations this architecture is able to be configured for, Butterworth, Chebyshev and Elliptic, and then some simulations are going to be presented.

**5.2.1 Butterworth.** The transfer function of a Butterworth filter is

$$H(s) = \frac{G_0}{\prod_{k=1}^n (s - s_k) / \omega_c}$$

5.3

The Butterworth polynomials may be written in complex form, as in Equation 5.3, but are usually written with real coefficients by multiplying pole pairs which are complex conjugates, such as  $s_1$  and  $s_n$ . The polynomials are normalized by setting  $\omega_c=1$ . The normalized Butterworth polynomials have the general form:

$$B_n(s) = \prod_{k=1}^{n/2} \left[ s^2 - 2s \cos \left\{ \left( \frac{2k + n - 1}{2n} \right) \pi \right\} + 1 \right]$$

5.4

If I choose the following parameters:

- $\omega_c = 2\pi \cdot 10^7$  rad/sec
- $n = 2$  (since it is a second order filter)

I obtain,

$$H(s) = \frac{3.948e^{15}}{s^2 + 8.886e^7 s + 3.948e^{15}}$$

5.5

Then, choosing  $C_1$  and  $C_2 = 0.1$  pF, and  $C_3 = 0$  pF, the other variables are:

- $G_{m1} = G_{m2} = G_{m4} = 6.2833 \cdot 10^{-6}$  Mhos
- $G_{m3} = 8.886 \cdot 10^{-6}$  Mhos

**5.2.2 Chebyshev.** The transfer function of a Chebyshev filter is

$$H(s) = \frac{1}{2^{n-1}\epsilon} \prod_{m=1}^n \frac{1}{(s - s_{pm})}$$

5.6

where,

$$\epsilon = \sqrt{10^{dB/10} - 1} \quad (\text{dB} = \text{ripple factor})$$

5.7

When it comes to low-pass filtering, the form of Equation 5.8 can also be used:

$$T_{LP}(s) = \frac{1}{(s/a_i\omega_c)^2 + (1/b_i)(s/a_i\omega_c) + 1}$$

5.8

where,

$$a_i = \left( \frac{1}{1 - h^2} - \sin^2 \theta_i \right)^{1/2}$$

5.9

$$b_i = \frac{1}{2} \left( 1 + \frac{1}{h^2 \tan^2 \theta_i} \right)^{1/2}$$

5.10

$$\theta_i = \frac{2i - 1}{n} \left( \frac{\pi}{2} \right) \quad 1 < i < \frac{n}{2}$$

5.11

$$h = \tanh \left( \frac{1}{n} \sinh \left( \frac{1}{\epsilon} \right)^{-1} \right)$$

5.12



If I choose the following parameters:

- $\text{dB} = 0.5 \text{ dB}$
- $\omega_c = 2\pi \cdot 10^7 \text{ rad/sec}$
- $n = 2$  (since it is a second order filter)

I obtain,

- $h = 0.70994$
- $\theta_i = 0 \text{ rad}$
- $a = 1.4199$
- $b = \infty$
- $e = 0.34931$

$$H(s) = \frac{5.986e^{15}}{s^2 + 8.957e^7 s + 5.986e^{15}}$$

5.13

Then, choosing  $C_1$  and  $C_2 = 0.1 \text{ pF}$ , and  $C_3 = 0 \text{ pF}$ , the other variables have the following values:

- $G_{m1} = G_{m2} = G_{m4} = 7.7369 \cdot 10^{-6} \text{ Mhos}$
- $G_{m3} = 8.957 \cdot 10^{-6} \text{ Mhos}$

**5.2.3 Elliptic.** The absolute value of the transfer function of an Elliptic filter is:

$$G_n(\omega) = \frac{1}{\sqrt{1 + \epsilon^2 R_n^2(\xi, \omega/\omega_0)}}$$

5.14

In this case I used the function that Matlab® has to build this kind of filters. In the other cases doing it manually was simple, but in this case I thought it would be easier using Matlab®'s function. The function is “[b,a] = ellip (n, Rp, Rs, Wn, 'low', 's')” where “b” is the polynomial at the numerator, and “a” is the polynomial at the denominator. “n” is the order, “Wn” is the cutoff frequency and its value (between 0 and 1) is the ratio between the cutoff frequency and the sampling frequency (even though is an analog function, Matlab works with sampled signals and hence, a sampling frequency is needed), “Rp” is the ripple factor for the passband and “Rs” is the ripple factor for the stopband. If I choose the following parameters:

- $\omega_c = 2\pi \cdot 10^7$  rad/sec
- $\omega_s = 2\pi \cdot 250 \cdot 10^6$  rad/sec
- $n = 2$  (since it is a second order filter)
- $R_p = 0.5$  dB
- $R_s = 30$  dB

I obtain,

$$H(s) = \frac{0.03163s^2 + 7.571e^{15}}{s^2 + 8.803e^7s + 6.049e^{15}}$$

5.15

Then, choosing  $C_1$  and  $C_2 = 0.1$  pF, the other variables have the following values:

- $G_{m1} = G_{m2} = 7.5565 \cdot 10^{-6}$  Mhos
- $G_{m4} = 8.005 \cdot 10^{-6}$  Mhos
- $G_{m3} = 8.803 \cdot 10^{-6}$  Mhos
- $C_3 = 3.163 \cdot 10^{-15}$  F

### 5.3 Verilog-A (Cadence) Model

The simulations performed in Cadence are just to show the functionality of the circuit and therefore, in order to avoid non-idealities no real circuits have been used. Instead, the components, like for instance the op-amps, have been taken from the library *ahdLib*, which contains ideal cells implemented using *Verilog-A* code.

**5.3.1 Transconductor.** The first block to be analyzed, and probably the simplest one, is the Transconductor stage. Since using real transistors to build the inverter would imply a deeper study and analysis than the one targeted, and because of the lack of other available components, I have used a switched voltage dependent current source as a Transconductor. Figures 5.8-5.10 show the schematic of the block used, the schematic used to simulate its own functionality, and the results of the transient simulation done of this block.

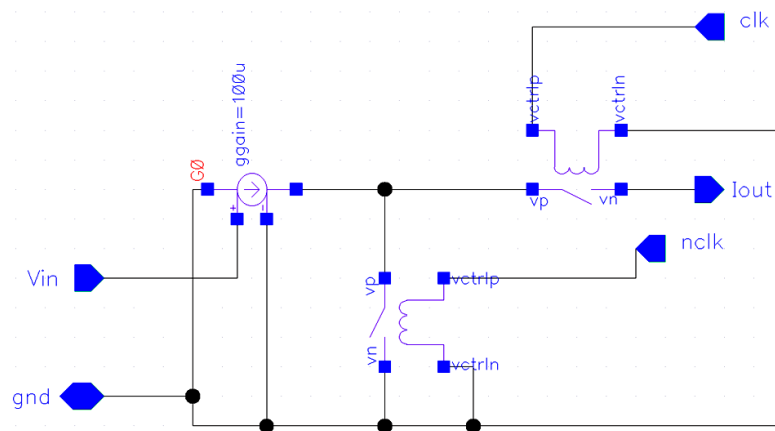


Figure 5.8. Transconductor in Cadence

For the simulation, the input signal used ( $V_{in}$  in Figure 5.9) was a sinusoid with amplitude equal to 1V and frequency 10 MHz. The resultant signal has the same amplitude and frequency that the input but is sampled at the sampling frequency (250 MHz). This is because for this simulation the constant of amplification of the dependent source is  $g_m=1$ . Note that the waveform seen at the output is named  $I_{out}$  whereas its units are Volts. The reasoning behind is because the waveform simulated is the voltage that the output resistor has at its terminals, but since the value of the resistor is 1 ohm the voltage plotted has the identical form that the current that goes through it. Another piece of information regarding the control signals of the switches, named “clk” and “nclk”, is: rise time = fall time = 1 ps; pulse width = 2 ns; period = 4 ns; duty-cycle = 50%.

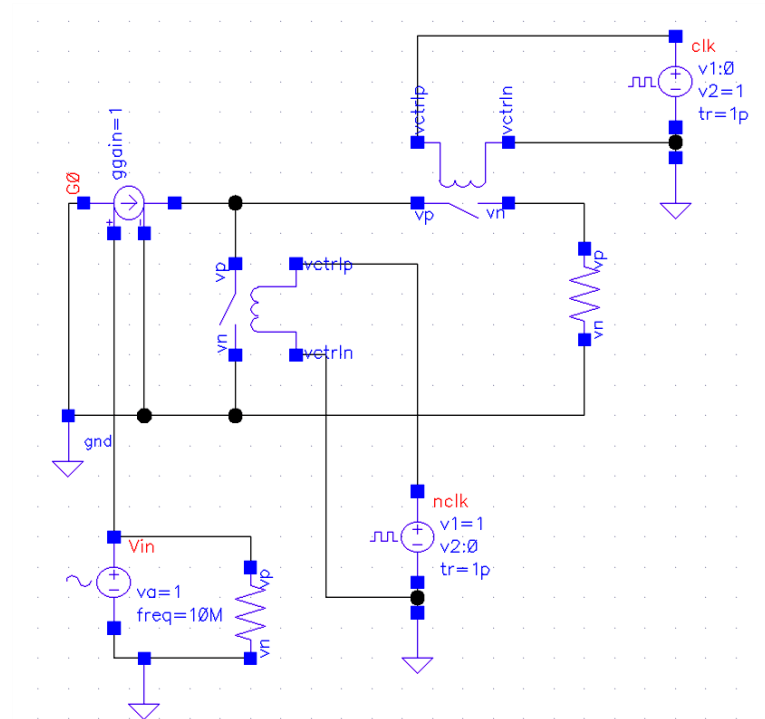


Figure 5.9. Schematic used for Transconductor simulation

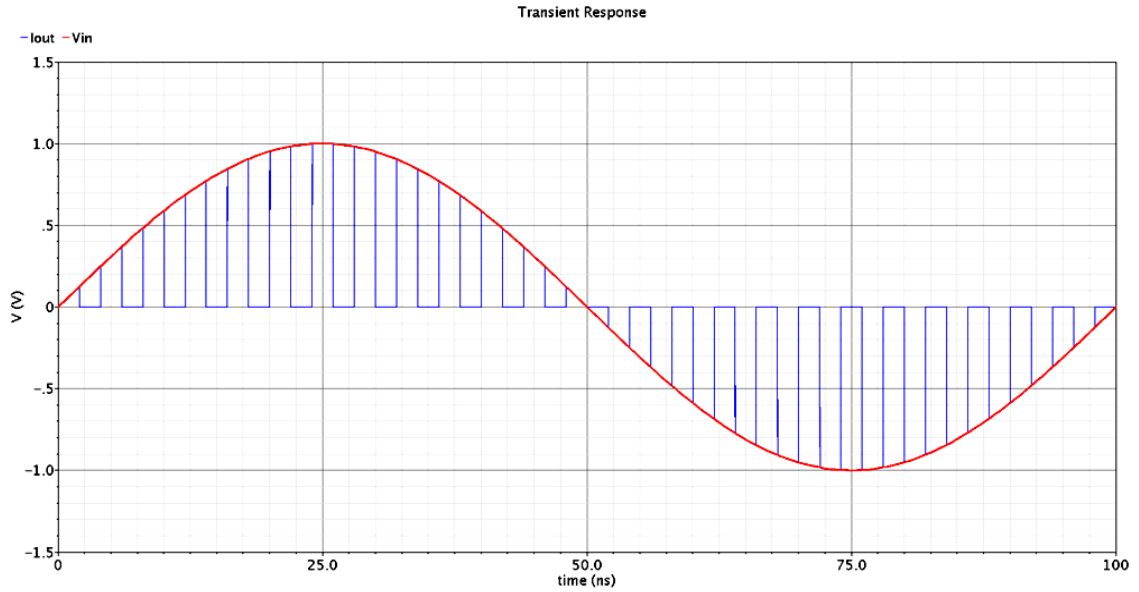


Figure 5.10. Transient response of the Transconductor

After having seen that the first block works properly, it is interesting to analyze the first stages which the signal is going to go through. Apart from the PGA1, the first step is the Anti-aliasing filter.

**5.3.2 Anti-aliasing filter.** As explained before, the Anti-aliasing filter is composed of a passive low-pass filter together with a 4-tap FIR-Gm filter (Sinc-Gm filter). Figures 5.11-5.12 in next page show the schematic of the whole circuit of the FIR filter. The 4-tap FIR-Gm filter has been simulated using the following parameters:

- The input signal ( $V_{in}$ ) has an amplitude of 1 mV and frequency equal to 10 MHz.
- The sampling signals (clk and nclk), that sample the output current, have the same as for the Transconductor: rise time = fall time = 1 ps; pulse width = 2 ns; period = 4 ns; duty-cycle = 50%.

- The four capacitors are used to simulate the input capacitance of the Transconductors used to store the value of the sampled signal (in [2] the Transconductors are inverters whose input capacitance is used as a register). The value chosen for them is  $C = 1 \text{ fF}$ .
- The resistors ( $1 \text{ G}\Omega$ ) used in parallel with these capacitors are necessary to allow Cadence to run the simulation and avoid errors due to initial conditions.
- The dependent current source have a transconductance of  $g_m = 0.25 \text{ S}$ , so that it produces a gain equal to 1 when the four currents are added together.
- The 4 control signals of the first four switches (clk1, clk2, clk3 and clk4) have the same properties that the control signals seen in Figure 5.9, with the difference that their pulse-width is 10 ps. That way this value is much lower than the period and helps the filter to achieve strongly attenuation of the noise where the notches are located.

The control signals can also be seen in Figure 5.12 where the signals at the top named clk1, clk2, clk3, clk4 control the four initial switches sampling the signal at 1 GHz with different phases. The other two signals showed at the middle and at the bottom of the figure, clk and /clk (nclk), control the two sampling switches placed at the end.

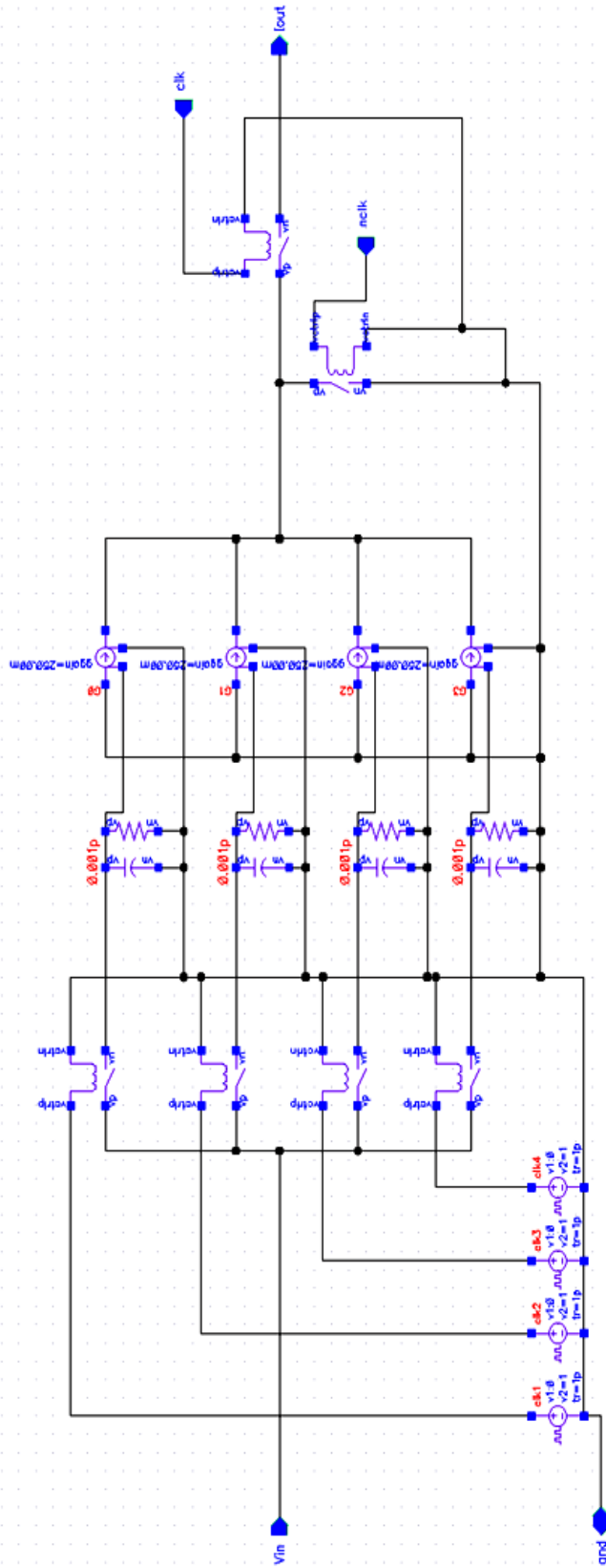


Figure 5.1.1. 4-tap FIR-Gm filter in Cadence

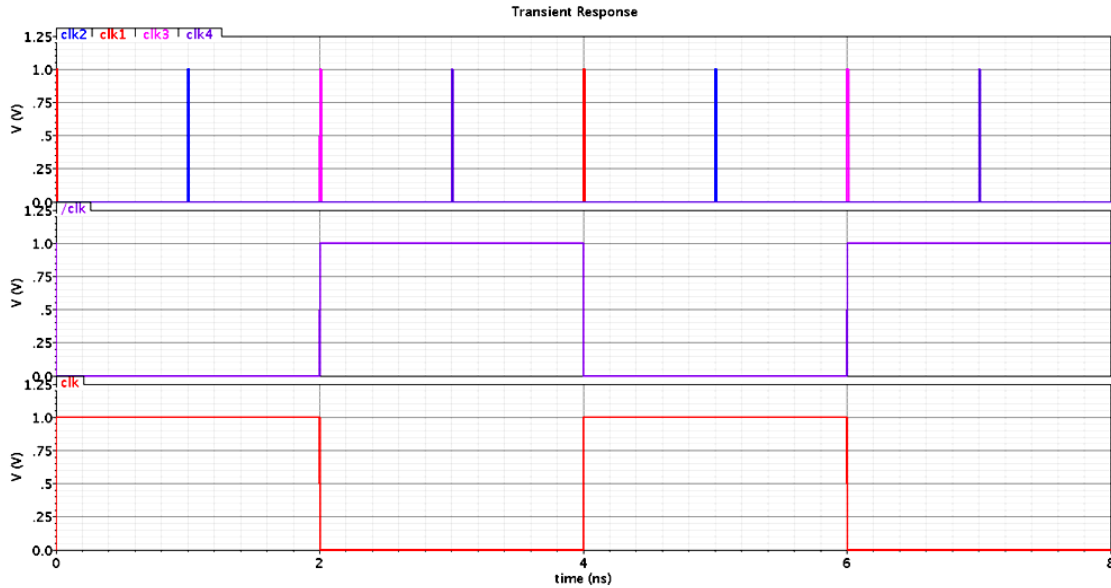


Figure 5.12. Control signals for the FIR filter

Figure 5.13 shows the transient simulation where the signals from all the nets in the circuit are plotted. At the top, there are the input signal and the four 1GHz-sampled signals, which can be distinguished from the input due to their form of stairs. These four signals are delayed between each other because their switches have different activation times (the delay between each switch is one clock period which is 1 ns). The signal at the bottom is the output current, which is the sum of the four sampled signals sampled afterwards for clk and nclk at 250 MHz.

As mentioned before, the 4-tap FIR-Gm filter is not good enough to be an Anti-aliasing filter by itself. Since the initial sampling is done at 1 GHz and the Sinc filter has only notches at 250 MHz, 500 MHz and 750 MHz, the noise located at 1 GHz would be folded onto baseband. To avoid this possibility, it is necessary a low-pass filter placed beforehand, which sufficiently attenuates the noises at such frequency. To see this effect a Simulink model has been first made (See Fig. 5.14).



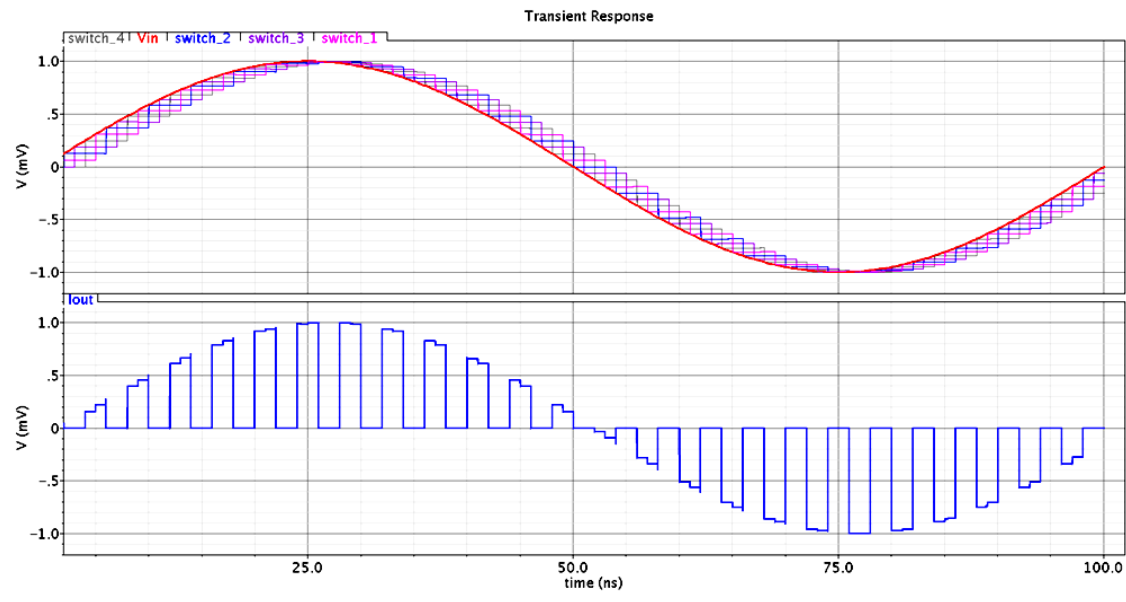


Figure 5.13. Transient response of the FIR filter

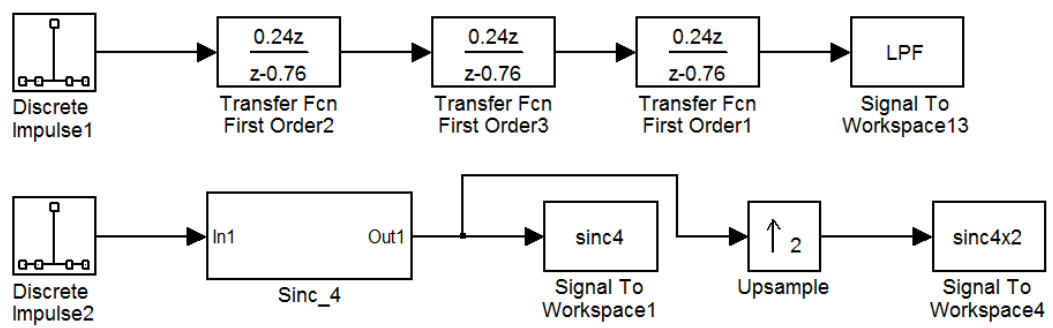


Figure 5.14. Anti-aliasing filter Simulink model

In Cadence, another four-order passive filter has also been created. The values used are not really important but to see the effect it causes within the joint response. Figure 5.15 shows the schematics of the passive filter. It is interesting having a look at the attenuation suffered at 1 GHz. Be aware that the units of the variables for the simulations related to the Anti-aliasing filter are expressed in dB10.

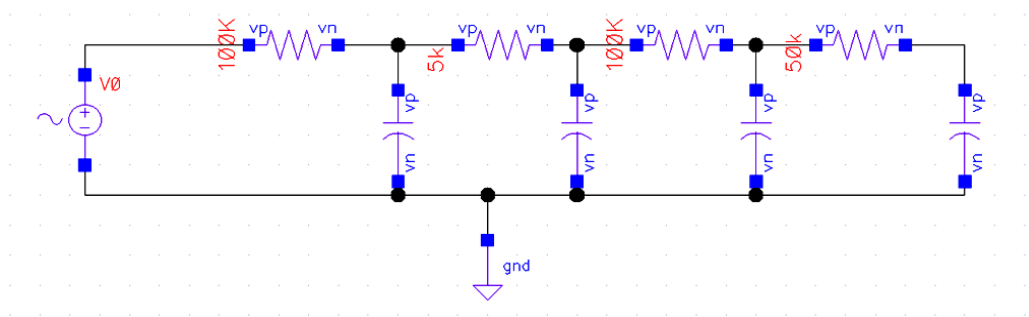


Figure 5.15. Passive filter for anti-aliasing

The values of the resistors can be seen in the picture while the capacitors are all of 0.1 pF. The AC response of the filter is displayed at Figures 5.16-5.17, showing a bandwidth of approximately 2 MHz. It is

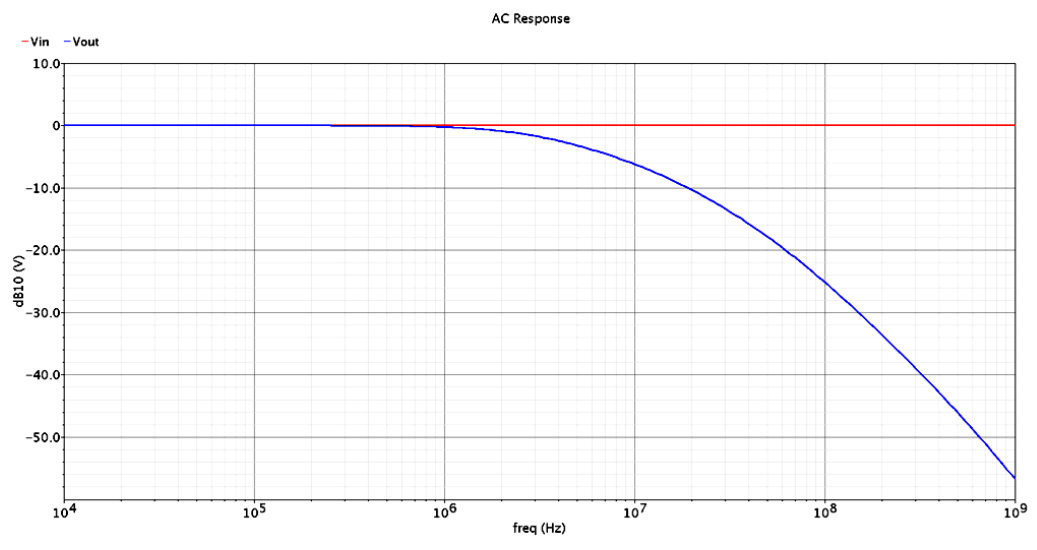


Figure 5.16. AC response of the passive filter\_1

To simulate the frequency response of the Sinc filter and of the complete Anti-aliasing filter a different method has necessarily been used. Due to the fact that the model has multiple inputs Cadence does not allow doing a sweep of all frequencies. For this

reason, multiple frequencies have been chosen. At each frequency simulations of the transient response have been done.

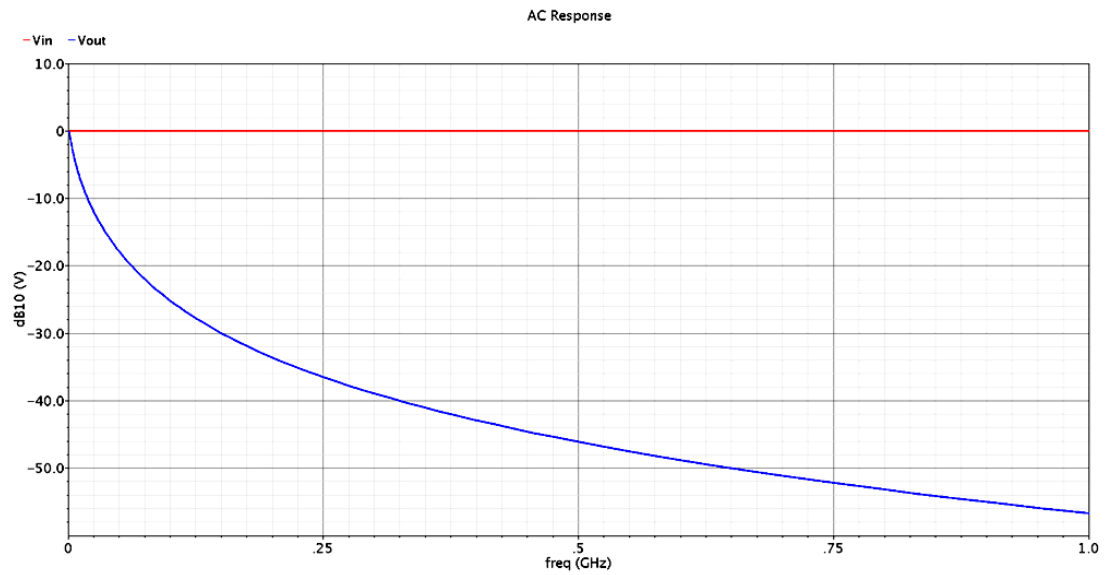


Figure 5.17. AC response of the passive filter\_2

Afterwards, several periods of the already stable signal have been used to calculate the Fast Fourier Transform (FFT), obtaining this way the value of the signal's amplitude. Figure 5.18 shows the circuit of the complete Anti-aliasing filter.

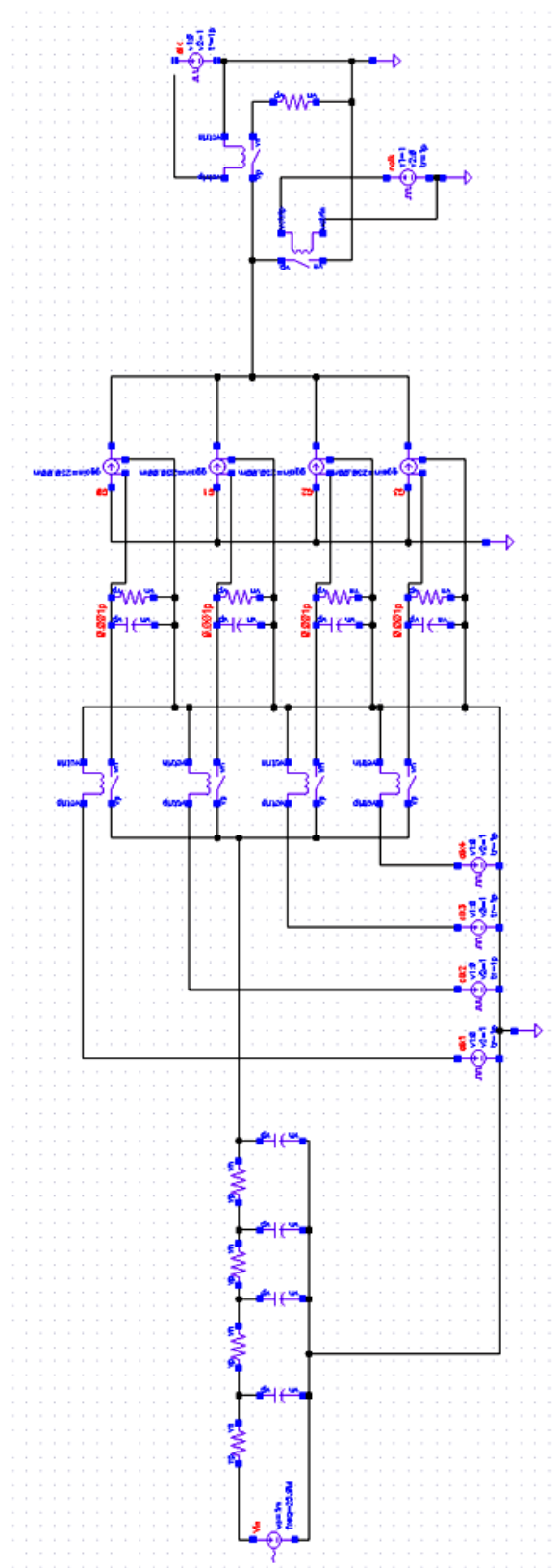


Figure 5.18. Anti-aliasing filter in Cadence

After modeling the Anti-aliasing filter, it is time to simulate the main part, which is composed for two identical and independent second-order DT-LPF (besides the first one includes the decimation stage). To do the simulation of the DT-LPF, though, it is necessary, first, to simulate all the different components that are in this filter, which are a Transconductor and an Integrator. The former has already been introduced, but not the latter. That is the reason why the following point is going to explain the Integrator.

**5.3.3 Integrator.** The Integrator is built by means of an op-amp and a capacitor as a negative feedback. For the feedback capacitor a value of 0.1 pF has been chosen. The output resistor of 1 M $\Omega$  is basically to have high output impedance and not to influence the transfer function of the circuit. The same purpose has the input resistor of 1  $\Omega$ . The supply of the op-amp (V1 and V2) is  $\pm 1V$  to allow an AC signal to go to positive and negative values. Higher values for the supply voltages are not necessary since the circuit is working with small signals. Figure 5.19 shows the schematic used to simulate the integrator.

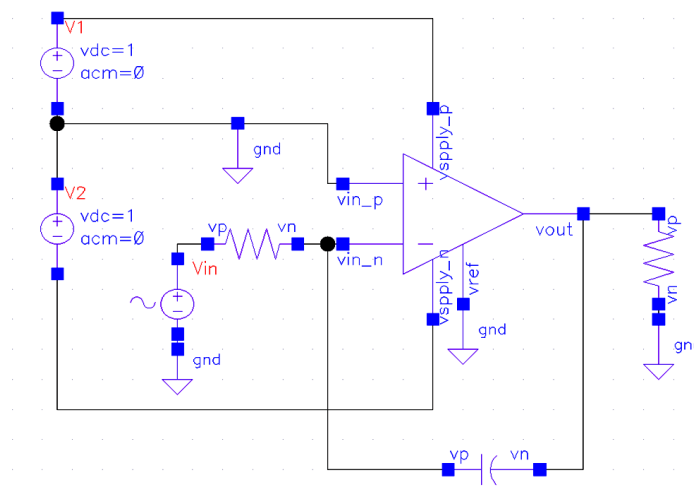


Figure 5.19. Schematic used for Integrator simulation

Regarding the op-amp there is another issue, and more important, to take into account: The Gain-Bandwidth product. Since the simulation is using ideal components the parameters of the op-amp have been set so that they do not limit either the amplification or the bandwidth of the circuit. However, circuits are not ideal and in future researching, when simulating a real circuit, the op-amp's characteristics are going to be an important factor to be aware of at this point. Figure 5.20 shows the AC response of the integrator with the values chosen.

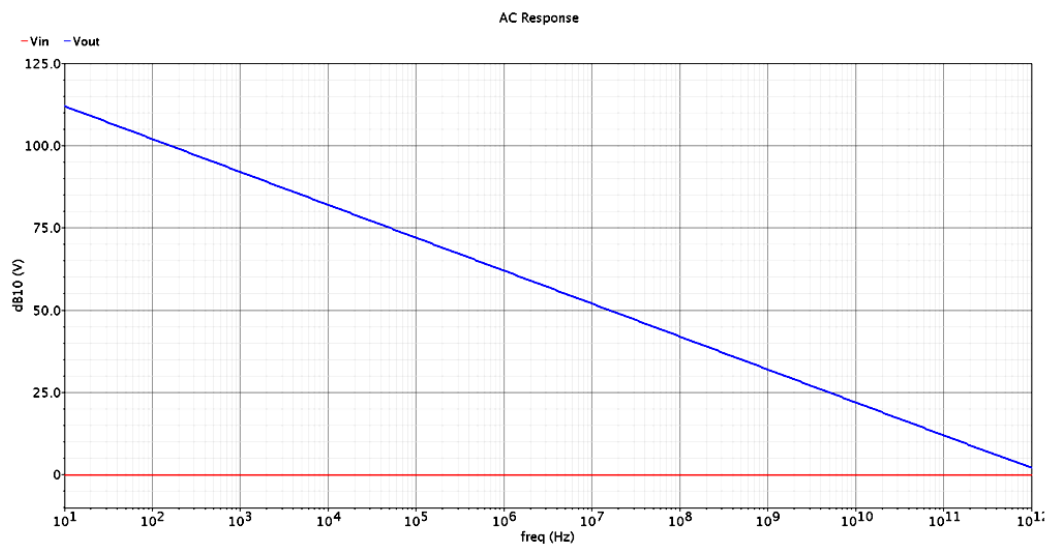


Figure 5.20. AC response of the Integrator

Another point that comes out when designing the integrator is the following. An Integrator like the ones used is also inverting. Therefore, it adds a negative sign that is being carried along the circuit. As seen in Figure 5.5, there exists an interleaving between negative and positive branches at every connection. This fact is the consequence of the two negative signs that the Integrators have. This interleaving is created to counteract them so that at the end there is just one effective negation, produced in the second

Integrator. The interleaving is an option when using differential architectures but in the Cadence model created it is not. That is the reasoning why two different models of Integrators have been used. The Inverting Integrator, which is placed right before the output (see Figure 5.21), and the Non-inverting Integrator (see Figure 5.22), in which a dependent source is used to create an ideal inversion that cancels out the negative sign introduced.

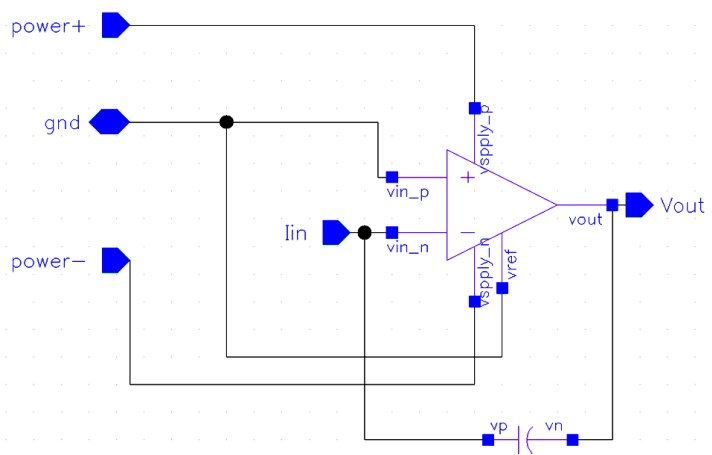


Figure 5.21. Inverting Integrator

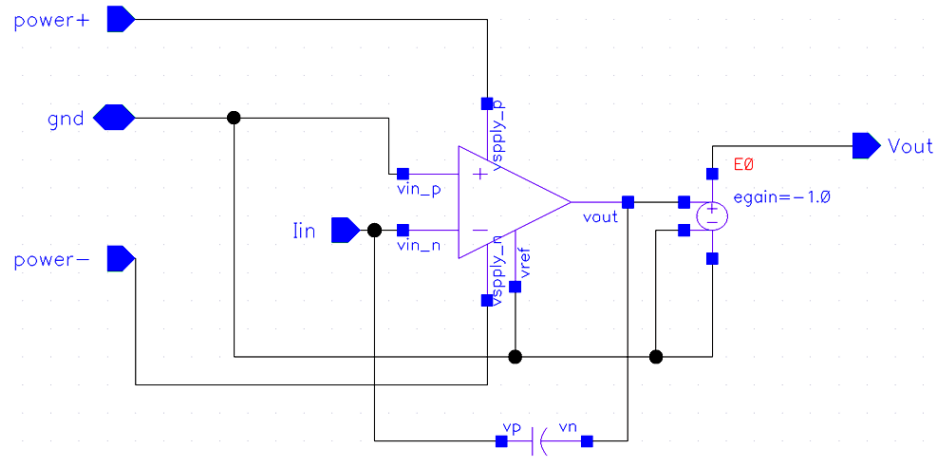


Figure 5.22. Non-inverting Integrator

**5.3.4 DT-LPF.** After having presented the integrator used in this circuit, it is time to simulate the 2<sup>nd</sup>-order DT-LPF. Figure 5.23 shows the schematic which is, as in Figure 5.5, mainly composed of six blocks, four Transconductors plus two Integrators. Each op-amp is supplied (V1 and V2) with the  $\pm 1$  V and each  $G_m$ 's sampling switches are controlled by one pair of control signals (ctr1-ctr8), whose pulse-width is the tool to modify the characteristics of the filter. The resistors ( $R_{\text{wire}} = 1 \Omega$ ) found within the feedback are simply a requisite so that the simulation can run and avoid shortcuts from to output to the input. For the simulation of the building block, as done previously for the Anti-aliasing filter, some transient simulations at several frequencies have been taken to calculate the FFT. The input signal's amplitude ( $V_{\text{in}}$ ) used to do that has been  $1 \mu\text{V}$ . Three simulations have been done for each configuration: Butterworth, Chebyshev and Elliptic. The specifications of each configuration are the ones calculated in the previous point. However, the values of the Transconductors are shared between the dependent current source and the control signals, in other words, the value of the transconductance  $G_m$  has been divided between  $g_m$  and the rate of the pulse-width over the period:

$$G_m = g_m \cdot \frac{T_u}{T}$$

5.16

where, if  $T$  and  $g_m$  are fixed, the equality is immediately solved. To calculate the values of  $T_u$  the resolution of the pulse generator has not been taken into account. Thus, taking into account that all the control signals in the schematic have period  $T = 4 \text{ ns}$  ( $f_s = 250 \text{ MHz}$ ), and that all current sources used have a  $g_m$  of  $10^{-4} \text{ S}$ , the other values are automatically solved for each configuration. Table 5.1 show the values for these variables used to tune the filter.



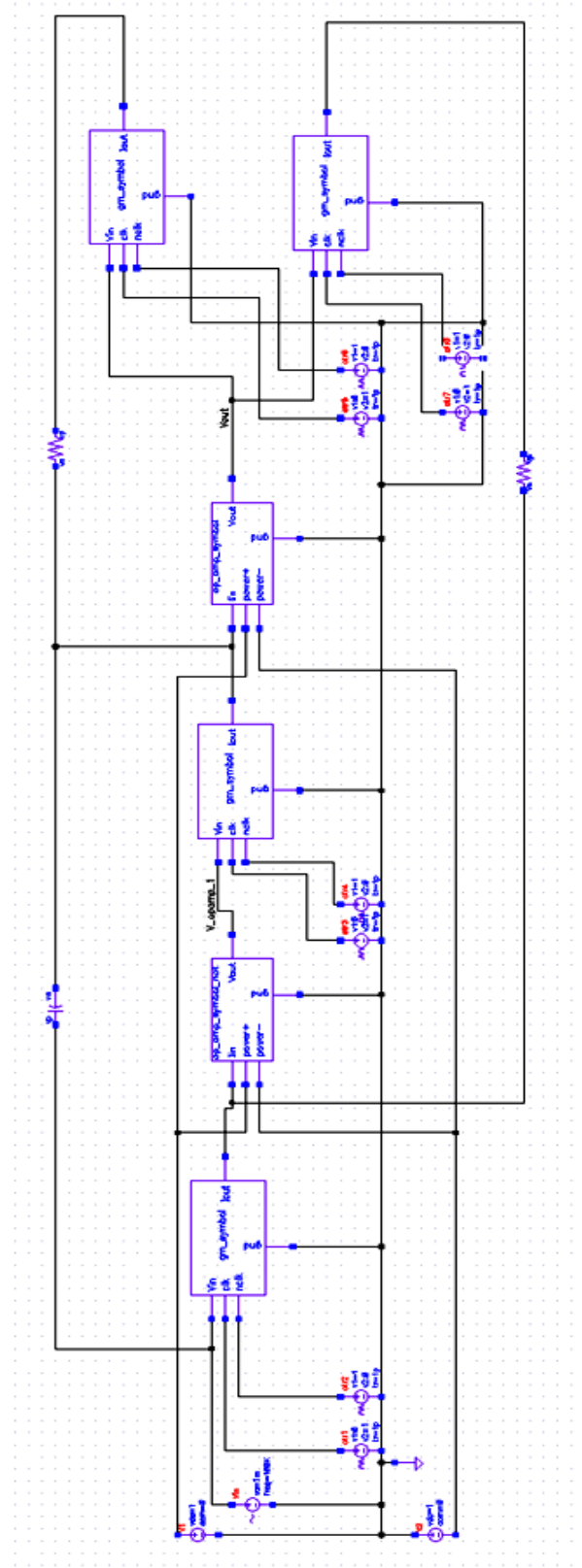


Figure 5.23. 2<sup>nd</sup>-order Discrete-time Low-pass filter model in Cadence

Table 5.1. Variables configuration for different types of filters

Variables	Configuration		
	Butterworth	Chebyshev	Elliptic
C3 (F)	0	0	3.163e-15
T <sub>u</sub> _gm1 (ps)	251.33	309.48	302.26
T <sub>u</sub> _gm2 (ps)	251.33	309.48	302.26
T <sub>u</sub> _gm3 (ps)	355.44	358.28	352.12
T <sub>u</sub> _gm4 (ps)	251.33	309.48	320.2

#### 5.4 Results

The first model to simulate was the Anti-aliasing filter. In this case, in order to compare the results of the Cadence model a Simulink one was first made. The simulations can be observed in Figure 5.24. It can be seen how the frequency response has notches at 250 MHz, 500 MHz and 750 MHz, and it achieves a good attenuation at 1 GHz.

For the simulation in Cadence, as explained before, the method used was different. The results of the simulation are showed in table 5.2 All the values in this table have been calculated based on the input signal's amplitude (equal to 1 mV) and multiplied by a factor of 2 in order to compensate the attenuation that the signal suffers due to the duty-cycle of 50% of the sampling clocks (V9 and V3).

If we plot the second column of table 5.2 together with the ideal form of a Sinc filter we obtain what can be seen in Figure 5.25.

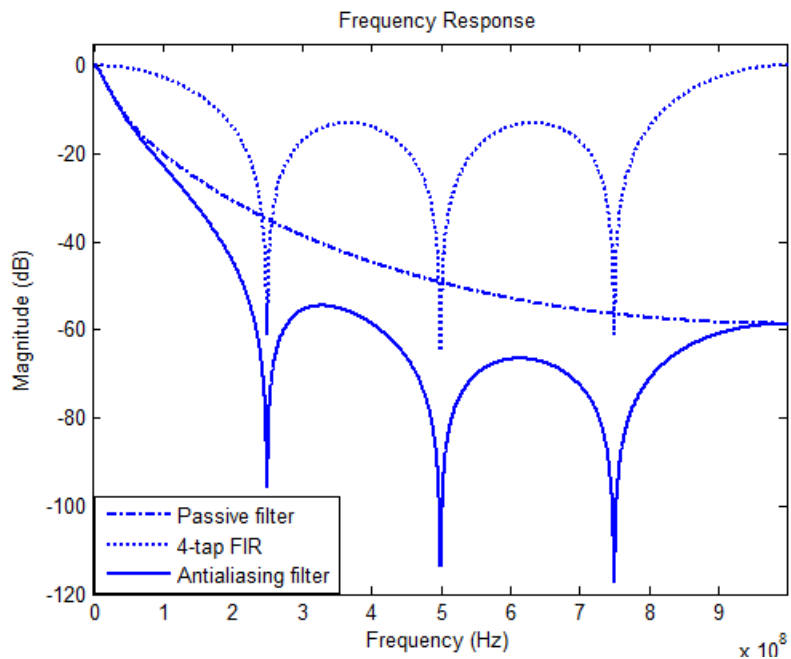


Figure 5.24. Frequency response of the Anti-aliasing filter

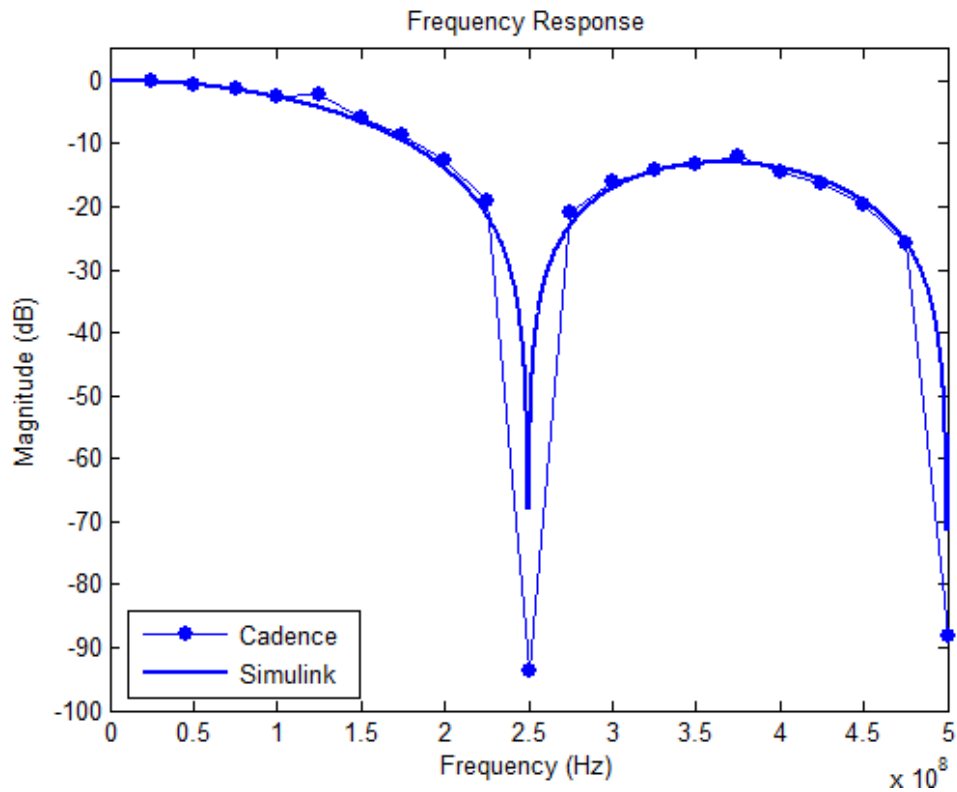


Figure 5.25. Cadence simulation of the 4-tap FIR-Gm filter

Table 5.2. Cadence simulations for the 4-tap FIR-Gm filter and the Anti-aliasing filter

Frequency	4-tap FIR-Gm filter (dB)	Passive LPF + FIR-Gm filter (dB)
2.50E+07	-0.1591	-24.1212
5.00E+07	-0.5712	-36.5547
7.50E+07	-1.3055	-45.4705
1.00E+08	-2.4122	-52.9824
1.25E+08	-2.2600	-55.3190
1.50E+08	-5.7660	-66.1656
1.75E+08	-8.7003	-72.6077
2.00E+08	-12.6140	-80.1339
2.25E+08	-19.1387	-89.6271
2.50E+08	-93.8285	-141.8636
2.75E+08	-20.9840	-96.8379
3.00E+08	-16.1506	-94.4627
3.25E+08	-14.0623	-94.6359
3.50E+08	-13.3005	-95.8468
3.75E+08	-11.9208	-98.7966
4.00E+08	-14.4825	-100.7670
4.25E+08	-16.3823	-104.4275
4.50E+08	-19.6623	-109.3410
4.75E+08	-25.7311	-116.8751
5.00E+08	-88.1566	-157.4624

The error introduced for the model is displayed in Table 5.3.

Table 5.3. Error introduced by the 4-tap FIR-Gm filter

Frequency	Verilog-A 4-tap FIR-Gm filter (dB)	Ideal 4-tap FIR-Gm filter(dB)	Error (dB) (Simulated-Ideal)
2.50E+07	-0.1591	-0.1535	-0.0056
5.00E+07	-0.5712	-0.6269	0.0557
7.50E+07	-1.3055	-1.4323	0.1268
1.00E+08	-2.4122	-2.6266	0.2144
1.25E+08	-2.2600	-4.2575	1.9975
1.50E+08	-5.7660	-6.4580	0.6920
1.75E+08	-8.7003	-9.4979	0.7976
2.00E+08	-12.6140	-13.8520	1.2380
2.25E+08	-19.1387	-21.3320	2.1933
2.50E+08	-93.8285	$-\infty$	$-\infty$
2.75E+08	-20.9840	-22.9030	1.9190
3.00E+08	-16.1506	-17.0500	0.8994
3.25E+08	-14.0623	-14.3920	0.3297
3.50E+08	-13.3005	-13.2080	-0.0925
3.75E+08	-11.9208	-13.0710	1.1502
4.00E+08	-14.4825	-13.8580	-0.6245
4.25E+08	-16.3823	-15.7070	-0.6753
4.50E+08	-19.6623	-19.0440	-0.6183
4.75E+08	-25.7311	-25.6140	-0.1171
5.00E+08	-88.1566	$-\infty$	$-\infty$

Although there are some values that differ from the ideal curve—the maximum error introduced is of 2.1933 dB—the overall form of the simulated case is really close to the ideal form. The errors' source could be several, starting with the calculation of Fourier's Transform, or simply because of the simulator itself.

The third column of Table 5.2 is the combination of the passive filter plus the Sinc filter, and is showed in Figure 5.26. The waveform obtained is similar to the one analyzed using Simulink. However, the attenuation is different because the passive filter itself is different.

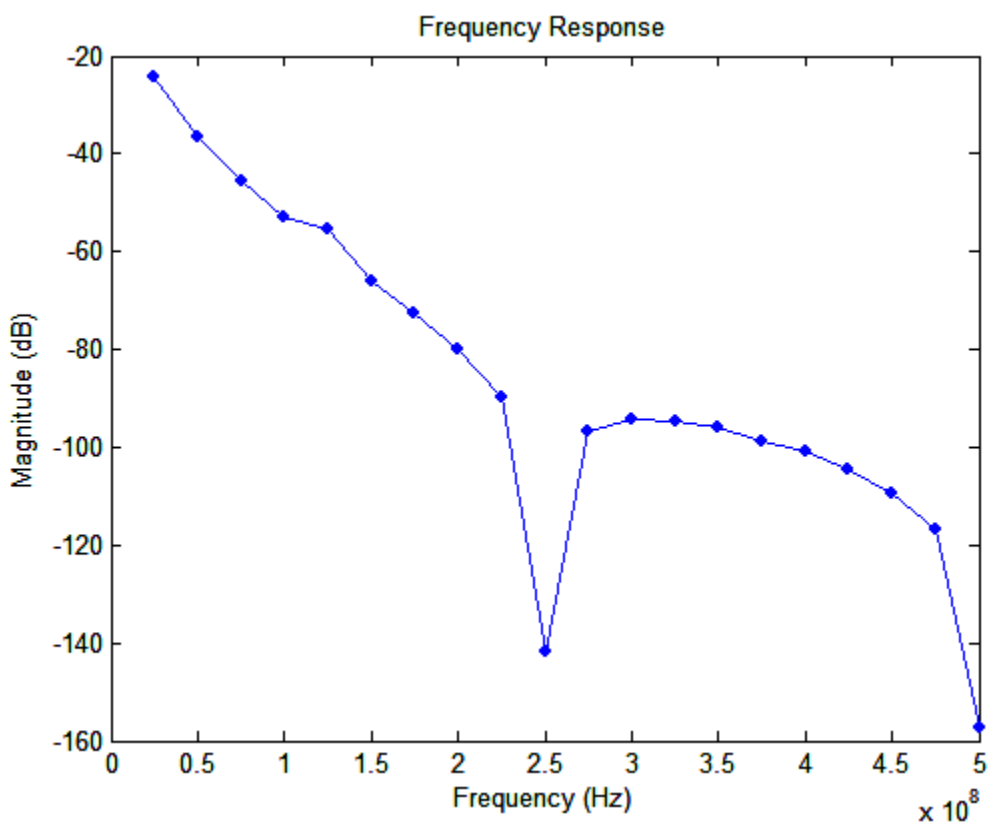


Figure 5.26. Cadence simulation of the Anti-aliasing filter

Table 5.4 shows the results obtained after simulating the DT-LPF for the three configurations, Butterworth, Chebyshev and Elliptic. If we compare the values in the table with the ideal values there exists some error. This error introduced for the model is quantified in Table 5.5.

Table 5.4. 2<sup>nd</sup>-order DT-LPF Cadence simulations for different configurations

Frequency	Magnitude (dB)		
	Butterworth	Chebyshev	Elliptic
1.00E+05	-0.4366	-0.2044	-0.7120
2.50E+05	-0.4366	-0.2034	-0.7110
5.00E+05	-0.4367	-0.1999	-0.7077
7.50E+05	-0.4388	-0.1941	-0.7021
1.00E+06	-0.4395	-0.1862	-0.6944
2.50E+06	-0.4659	-0.0949	-0.6051
5.00E+06	-0.7439	0.1550	-0.3581
7.50E+06	-1.7483	0.2718	-0.5071
1.00E+07	-3.4322	-0.2233	-0.7340
2.50E+07	-16.6532	-12.2415	-13.9119
5.00E+07	-29.4567	-25.4046	-34.2988
7.50E+07	-38.2671	-34.2180	-37.6431
1.00E+08	-46.9861	-42.7290	-32.2046
1.25E+08	-66.4731	-60.3409	-30.7153

Table 5.5. 2<sup>nd</sup>-order DT-LPF error for different configurations

Frequency	Error (dB) (Simulated-Ideal)		
	Butterworth	Chebyshev	Elliptic
1.00E+05	-0.4366	-0.2053	-0.2130
2.50E+05	-0.4366	-0.2046	-0.2120
5.00E+05	-0.4366	-0.2048	-0.2127
7.50E+05	-0.4386	-0.2049	-0.2131
1.00E+06	-0.4390	-0.2054	-0.2134
2.50E+06	-0.4486	-0.2079	-0.2151
5.00E+06	-0.4599	-0.2160	-0.2251
7.50E+06	-0.4883	-0.1902	-0.4702
1.00E+07	-0.3722	-0.2056	-0.1960
2.50E+07	-0.6532	-0.4415	-0.4119
5.00E+07	-1.4567	-1.2046	-2.7988
7.50E+07	-3.2671	-2.9180	6.8569
1.00E+08	-6.9861	-6.3290	2.8954
1.25E+08	-22.6731	-20.1409	2.2847

The representation of the results, for each configuration, plotted together with the ideal response is done in Figures 5.27-5.33.



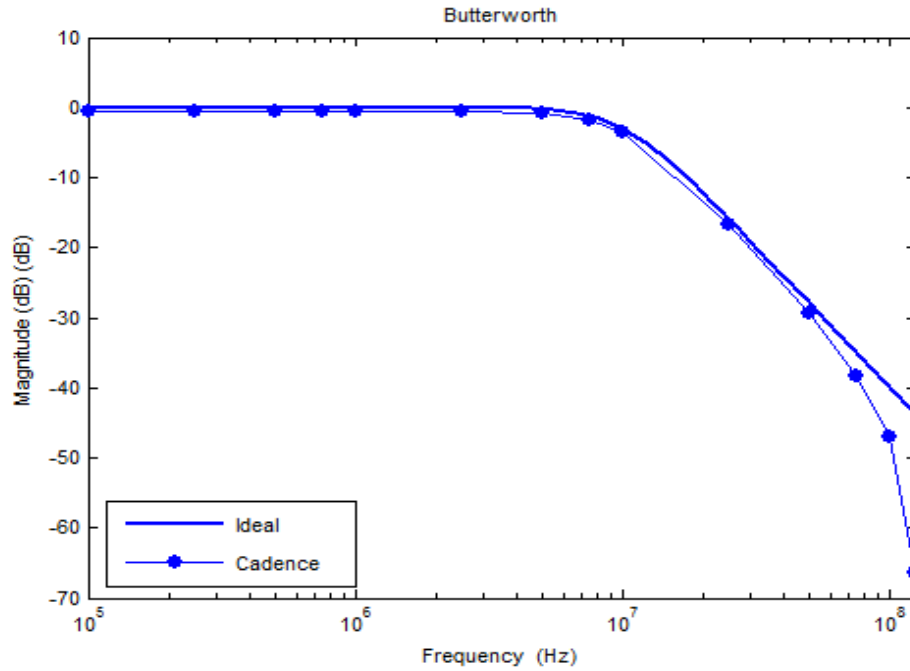


Figure 5.27. Comparison between the DT-LPF Cadence model and the ideal response (Butterworth) (1)

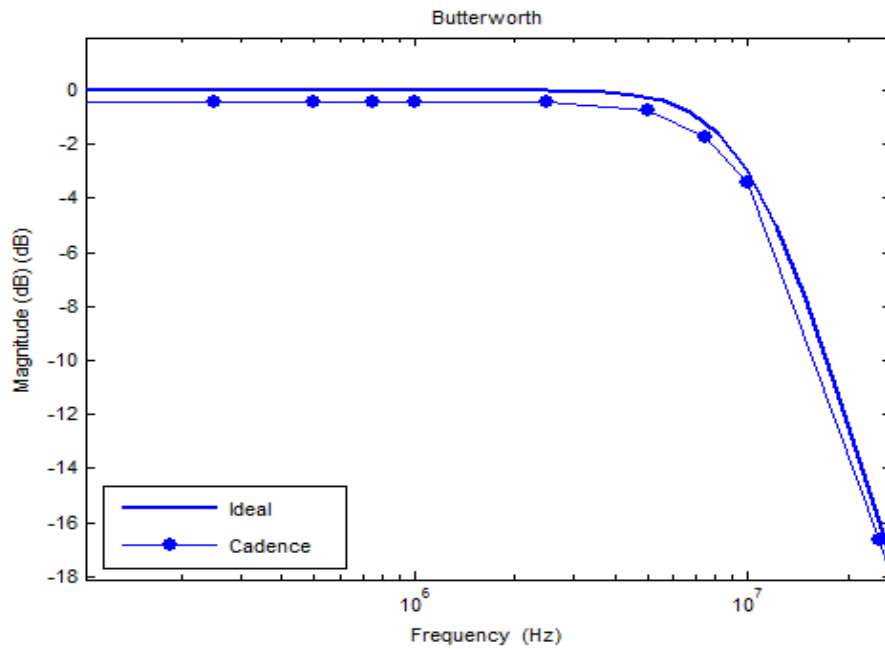


Figure 5.28. Comparison between the DT-LPF Cadence model and the ideal response (Butterworth) (2)

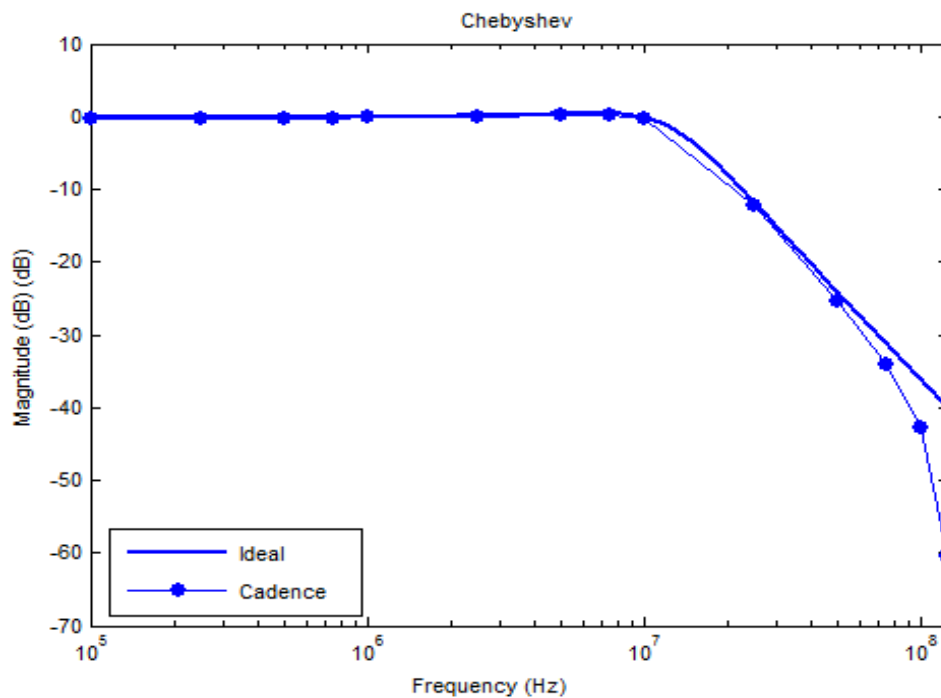


Figure 5.29. Comparison between the DT-LPF Cadence model and the ideal response (Chebyshev) (1)

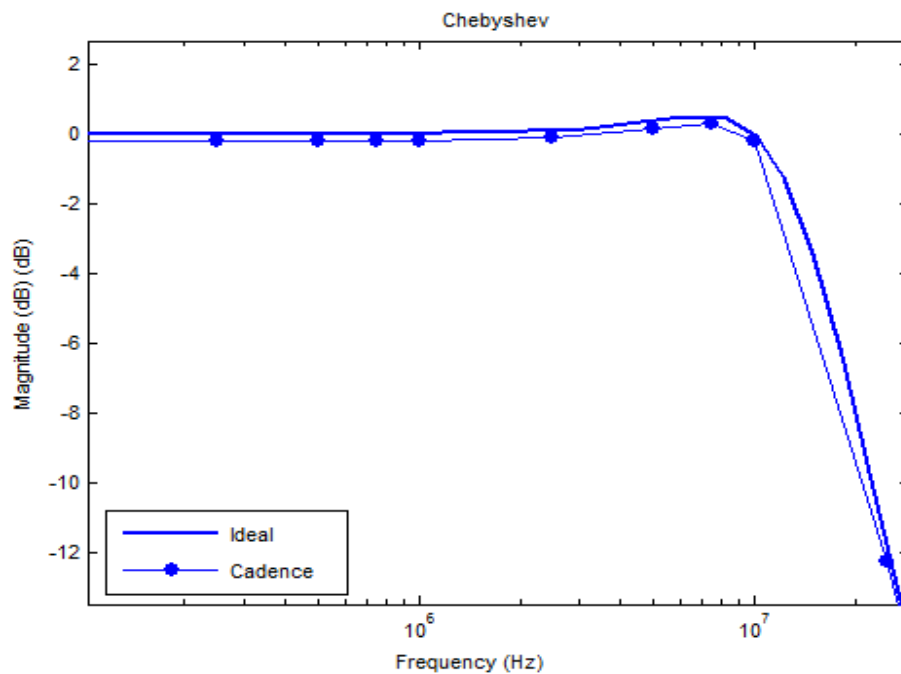


Figure 5.30. Comparison between the DT-LPF Cadence model and the ideal response (Chebyshev) (2)

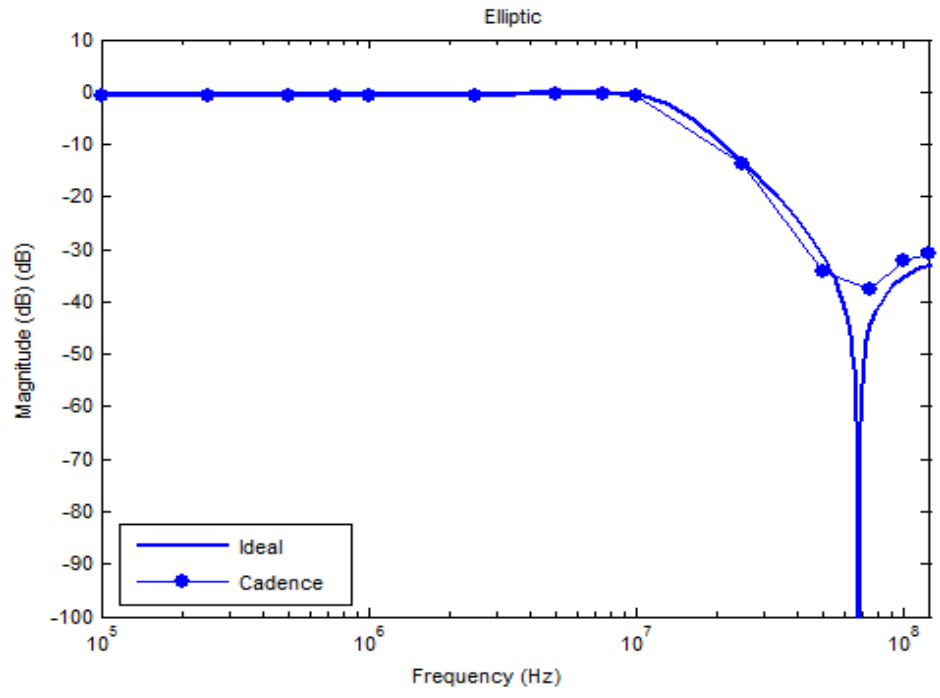


Figure 5.31. Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (1)

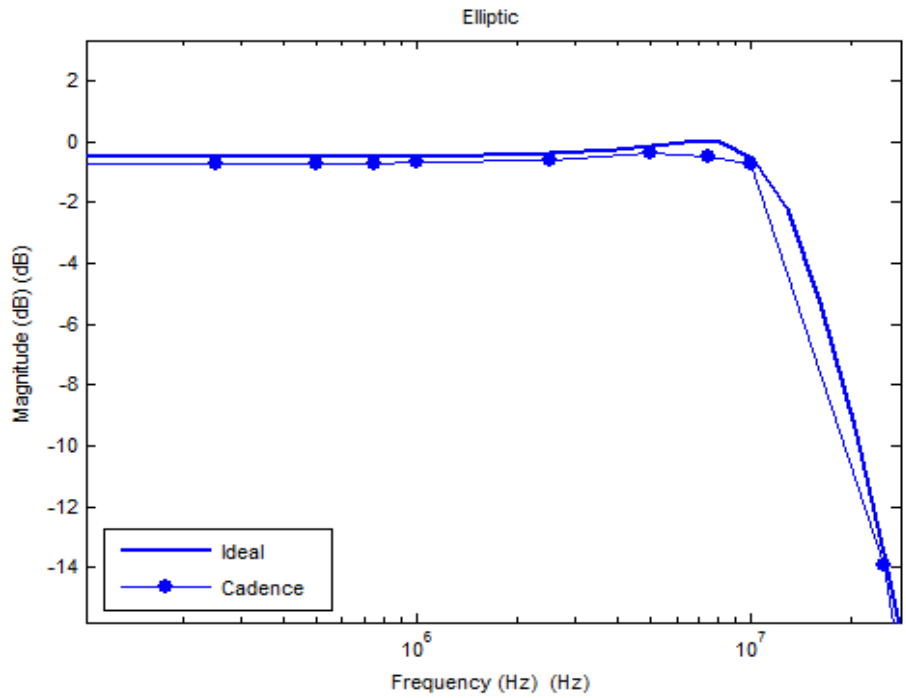


Figure 5.32. Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (2)

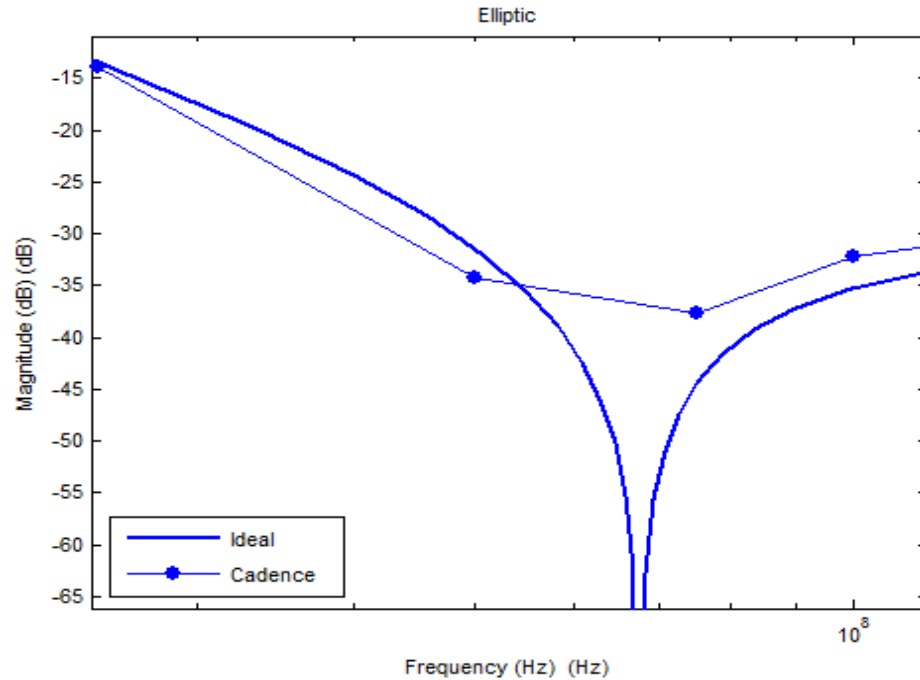


Figure 5.33. Comparison between the DT-LPF Cadence model and the ideal response (Elliptic) (3)

## 5.5 Conclusions

The objective of this study was to analyze the behavior of a widely tunable filter for Software-Defined Radio. To do so, I have resorted to the simulation of the circuit by using Verilog-A modules in Cadence.

After having plotted the simulations performed in Cadence together with the ideal curve, it is relevant the fact that the resulting data have been successful, being the waveform obtained close to the ideal one within an accepting error, and behaving accordingly to the function it has been designed for. Next studies related to the same circuit will have in the Cadence model presented a reliable model to compare their results and it is a starting point from where a deeper study to physically build the circuit can be done.

## CHAPTER 6

### CONCLUSIONS

The purpose of this thesis project is the modeling of reconfigurable sampling receivers and baseband for Software-Defined Radio. An historic review of the classic review was first made to present their differences compared to the sampling ones. Second, a deep study of the tools used for the Discrete-time Signal Processing was displayed so that each tool's functionality was understood. Afterwards, two receivers, from [7] and [10], were compared. For the second one, a model in Simulink was created. The model includes a transceiver and the front-end part of the receiver. In the case tested, a basic BPSK modulation was used because it would easily show graphically its behavior. Since the model is independent of the modulation used, other modulations can be used and tested. Therefore, the file presented can be used in the future for testing and also as a basis to build other receivers for different purposes.

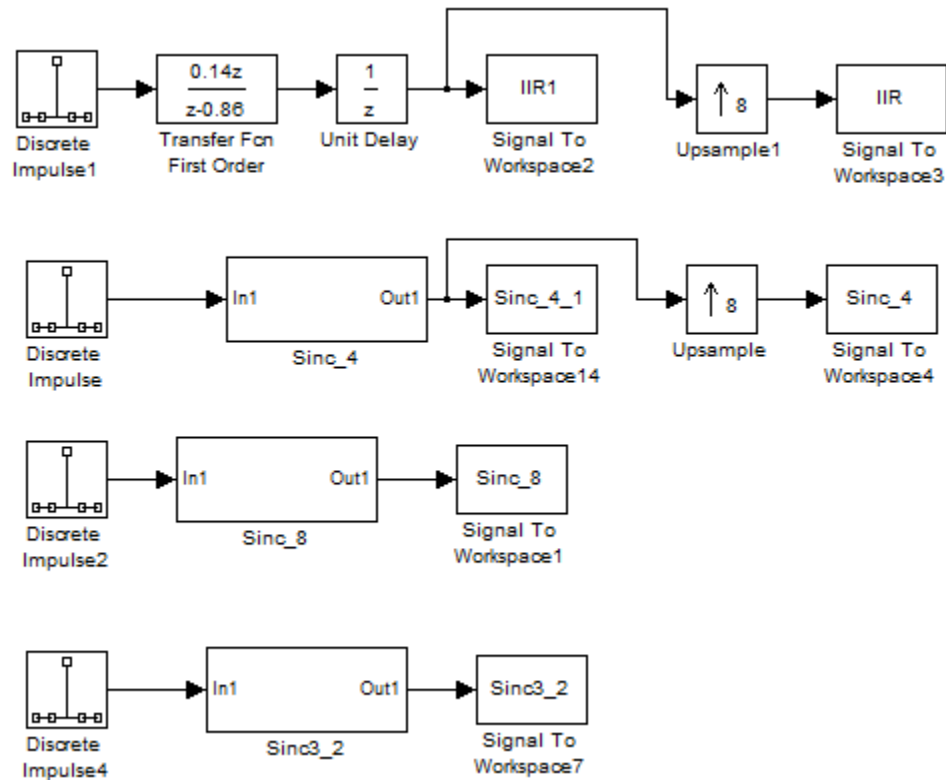
The last part of the thesis narrowed down the topic under study, and a tunable filter, from paper [2], was analyzed. This time the tool used for the simulation was Cadence. The model, done in Verilog-A, displayed good results, and the error introduced was in most of the frequency range less than 0.5 dB. The error increased within the stopband, when the attenuation introduced was higher than 30 dB, probably due to the lack of resolution of Cadence. This model is reliable for future studies and it means the first step for further modeling and research, and for building the real circuit.

## APPENDIX A

MATLAB CODE TO OBTAIN THE MTDSM IMPULSE RESPONSE

## A.1 MTDSM (Simulink)

To obtain the frequency response of all the filters used in the MTDSM's Simulink file I used the next model.



Figure(1)

```
f=2400000/4096*(0:4095);
plot(f,10*log(abs(fft(Sinc_8(1,:),4096))))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(2)

```
f=300000/4096*(0:4095);
plot(f,10*log(abs(fft(IIR1,4096))))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(3)

```
f=300000/4096*(0:4095);
plot(f,10*log(abs(fft(Sinc_4_1(1,:),4096))))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(4)

```
f=75000/4096*(0:4095);
plot(f,10*log(abs(fft(Sinc3_2(1,:),4096))))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(5)

```
f=2400000/4096*(0:4095);
plot(f,10*log(abs(fft(IIR(1,:),4096))))
hold on
plot(f,10*log(abs(fft(Sinc_4,4096))))
hold on
plot(f,10*log(abs(fft(Sinc_8(1,:),4096))))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(6)

```
mtdsm=fft(IIR',4096).*fft(Sinc_4(1,:),4096).*fft(Sinc_8(1,:),4096);
plot(f,10*log(abs(mtdsm)))
title('Frequency Response'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

## A.2 Receiver (Simulink)

Figure(1)

```
subplot(3,2,[1 2]);
t=0.00005/size(baseband,1)*(0:size(baseband,1)-1);plot(t,real(baseband(:,1)));
title('Baseband signal'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,3);
t=0.00005/size(modulated,3)*(0:size(modulated,3)-1); plot(t,real(modulated(1,:)));
title('Signal Modulated'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,5);
t=0.00005/size(RF_signal,3)*(0:size(RF_signal,3)-1); plot(t,real(RF_signal(1,:)));
title('Signal Received'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,4); f=24000000000/4096*(0:4095); M = fft(modulated(1,:),4096);
Pmm = M.*conj(M)/4096; plot(f,10*log(Pmm));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

subplot(3,2,6); f=24000000000/4096*(0:4095); R = fft(RF_signal(1,:),4096);
Prf = R.*conj(R)/4096; plot(f,10*log(Prf));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(2)

```
subplot(3,2,1); t=0.00005/size(sampled,3)*(0:size(sampled,3)-1); plot(t,sampled(1,:));
title('Signal sampled (In-Phase)'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,3); t=0.00005/size(sinc_8,3)*(0:size(sinc_8,3)-1); plot(t,sinc_8(1,:));
```



```

title('Sinc_8-Filter output'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,5); t=0.00005/size(down_8,3)*(0:size(down_8,3)-1); plot(t,down_8(1,:));
title('Signal downsampled (N=8)'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,2);
f=2400000000/4096*(0:4095); plot(f,20*log(abs(fft(sampled(1,)/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

subplot(3,2,4);
f=2400000000/4096*(0:4095); plot(f,20*log(abs(fft(sinc_8(1,)/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

subplot(3,2,6);
f=3000000000/4096*(0:4095); plot(f,20*log(abs(fft(down_8(1,)/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

Figure(3)
subplot(3,2,1); t=0.00005/size(down_8,3)*(0:size(down_8,3)-1); plot(t,down_8(1,:));
title('Signal downsampled (N=8)'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,3); t=0.00005/size(sinc_4,1)*(0:size(sinc_4,1)-1); plot(t,sinc_4);
title('Sinc_4-Filter output'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,5); t=0.00005/size(down_4,1)*(0:size(down_4,1)-1); plot(t,down_4);
title('Signal downsampled (N=4)'); xlabel('Time (s)'); ylabel('Magnitude (V)');

subplot(3,2,2);
f=3000000000/4096*(0:4095); plot(f,20*log(abs(fft(down_8(1,)/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

subplot(3,2,4);
f=3000000000/4096*(0:4095); plot(f,20*log(abs(fft(sinc_4/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');

subplot(3,2,6);

```

```
f=75000000/4096*(0:4095); plot(f,20*log(abs(fft(down_4/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

Figure(4)

```
subplot(3,2,1); t=0.00005/size(down_4,1)*(0:size(down_4,1)-1); plot(t,down_4);
title('Signal downsampled (N=4)'); xlabel('Time (s)'); ylabel('Magnitude (V)');
```

```
subplot(3,2,3); t=0.00005/size(sinc3_2,1)*(0:size(sinc3_2,1)-1); plot(t,sinc3_2);
title('Sinc^3_2-Filter output'); xlabel('Time (s)'); ylabel('Magnitude (V)');
```

```
subplot(3,2,5); t=0.00005/size(down_2,1)*(0:size(down_2,1)-1); plot(t,down_2);
title('Signal downsampled (N=2)'); xlabel('Time (s)'); ylabel('Magnitude (V)');
```

```
subplot(3,2,2);
f=75000000/4096*(0:4095); plot(f,20*log(abs(fft(down_4/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

```
subplot(3,2,4);
f=75000000/4096*(0:4095); plot(f,20*log(abs(fft(sinc3_2/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

```
subplot(3,2,6);
f=37500000/4096*(0:4095); plot(f,20*log(abs(fft(down_2/sqrt(4096),4096))));
title('Power Spectral Density'); xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
```

## BIBLIOGRAPHY

- [1] Darabi, H. and Abidi, A. "A 4.5mW 900-MHz CMOS Receiver for Wireless Paging." *IEEE Journal of Solid-State Circuits*. 35 (August. 2000): 1085-1096.
- [2] Kitsunezuka, M., Hori, S., and Maeda, T. "A Widely-Tunable, Reconfigurable CMOS Analog Baseband IC for Software-Defined Radio." *IEEE Journal of Solid-State Circuits*. 44.9 (September 2009): 2496-2502.
- [3] Koh, J., Muhammad, K., Staszewski, B., Gomez, G. and Horoun, B. "A Sigma-Delta ADC with a built-in Anti-aliasing filter for Bluetooth receiver in 130nm digital process." *IEEE Custom Integrated Circuits Conference*. 25-6 (October 2004).
- [4] Lombardi, A., Bonizzoni, E., Malcovati, P. and Maloberti, F. "A Low Power Sinc<sup>3</sup> Filter for  $\Sigma\Delta$  Modulators." *Proc. of 2007 International Symposium on Circuit and Systems (ISCAS)*. (May 2007): 4008-4011.
- [5] Milić, L. "Multirate Filtering for Digital Signal Processing: Matlab® Applications." University of Belgrade, Serbia.
- [6] Muhammad, K., Leipold, D., Staszewski, B., Ho, Y.-C., Hung, C. M., Maggio, K., Fernando, C., Jung, T., Wallberg, J., Koh, J.-S., John, S., Deng, I., Moreira, O., Staszewski, R., Katz, R. and Friedman, O. "A Discrete-Time Bluetooth Receiver in a 0.13 $\mu$ m Digital CMOS Process." *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*. 527 (February 2004): 268-269.
- [7] Shen, D.H., Chien-Meen Hwang, Lusignan, B.B. and Wooley, B.A. "A 900-MHz RF Front-End with Integrated Discrete-Time Filtering." *IEEE Journal of Solid-State Circuits*. 31.12 (December 1996): 1945-1954.
- [8] Staszewski, R.B., Muhammad, K. and Leipold, D. "Digital RF Processor Techniques for Single-Chip Radios (Invited)." *IEEE Custom Integrated Circuits Conference 2006*. 10-13 (September 2006): 789-796.
- [9] Staszewski, R. B., Muhammad, K., and Leipold, D. "Digital RF Processor (DRPTM) for Cellular Phones." *IEEE ISCAS*. (2005).
- [10] Staszewski, R.B., Muhammad, K., Leipold, D., Chih-Ming Hung, Yo-Chuol Ho, Wallberg, J.L., Fernando, C., Maggio, K., Staszewski, R., Jung, T., Jinseok Koh, John, S., Irene Yuanying Deng, Sarda, V., Moreira-Tamayo, O., Mayega, V., Katz, R., Friedman, O., Eliezer, O.E., de-Obaldia, E. and Balsara, P.T. "All-Digital TX Frequency Synthesizer and Discrete-Time Receiver for Bluetooth Radio in 130-nm CMOS." *IEEE Journal of Solid-State Circuits*. 39.12 (December 2004): 2278-2291.