

CHALMERS



Design of a Down Converter for a Galileo Receiver

Master of Science Thesis

Alexander Vickberg

Yue Wu

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2010

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Design of a down converter for Galileo receiver

Alexander Vickberg
Yue Wu

© Alexander Vickberg, June 2010.
© Yue Wu, June 2010.

Examiner: Sven Knutsson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2010

ABSTRACT

This report presents the work performed in the master thesis work "Design of a Down Converter for a Galileo Receiver". The thesis work has designed a receiver front-end down converter for the new European navigation system, Galileo. For this thesis work, the development platform Neptune-V5 VXS from Tekmicro has been used, which offers a high-speed RF sampling analogue-to-digital converter together with high-speed Xilinx Virtex-5 FPGAs. A digital down converter architecture utilizing direct RF sampling techniques is defined in this thesis work. The work also includes modeling the FPGA design in MATLAB, implementing this design in an FPGA using VHDL and targeting an ASIC implementation considering size/power constraints. Finally, through laboratory tests supported by analysis and/or system modeling the performance of key parameters have been tested.

(This page is intentionally left blank)

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to our main supervisors Jacob Christensen and Thomas Lindgren at RUAG space.

To Jacob for giving us the opportunity to do this project. He always trusts us and has always been there to encourage us when we found ourselves in a difficult phase. His door has always been open to us in spite of his busy schedule. It has been a very inspirational experience to work with him.

To Thomas for guiding us through the project with his experiences. He has been supporting us with his suggestions and ideas of the design and helped us structure and write the report. He has always been patient to our questions no matter how trivial they were. The lessons we learnt from him are valuable for our future engineer life.

Special thanks to our examiner Sven Knutsson from the department of Computer science and engineering, Chalmers for giving feedback and helping us improve our writing.

Finally we would like to thank Chalmers University of Technology for accepting us as Master students and RUAG space for allowing us working in a positive and collaborative environment.

Göteborg, 7 June 2010
Alexander Vickberg
Yue Wu

(This page is intentionally left blank)

Document Organization

- Chapter 1 gives the purpose and scope of this thesis, overview of Galileo system
- Chapter 2 introduces the literature and required information used in this thesis work
- Chapter 3 defines the baseline architecture solution and detailed algorithm design
- Chapter 4 shows the MATLAB model used in the design and verification
- Chapter 5 presents the detailed hardware implementation
- Chapter 6 contains the verification methodology and results
- Chapter 7 analysis the noise error in ADC, SNR for down converter and FPGA budgets
- Chapter 8 summarizes the conclusions

(This page is intentionally left blank)

Abbreviations and Acronyms

| | |
|----------|--|
| ADC | Analog-to-digital converter |
| ASIC | Application specific integrated circuit |
| DDR | Double data rate |
| DEMUX | Demultiplexer |
| DSP | Digital signal processing |
| f_0 | Fundamental reference frequency, 1.023 MHz |
| f_c | Signal carrier center frequency |
| f_s | Sampling frequency |
| f_{LO} | Local oscillator frequency |
| FIR | Finite impulse response |
| FPGA | Field programmable gate array |
| GB | Gigabyte |
| GNSS | Global navigation satellite system |
| GPS | Global positioning system |
| LSB | Least significant bit |
| LUT | Look-up table |
| MSB | Most significant bit |
| NCO | Numerically controlled oscillator |
| PC | Personal computer |
| RF | Radio frequency |
| SDRAM | Synchronous dynamic random access memory |
| SFDR | Spurious-free dynamic range |
| SNR | Signal to Noise ratio |
| VHDL | VHSIC hardware description language |
| VHSIC | Very high speed integrated circuit |

(This page is intentionally left blank)

TABLE OF CONTENTS

PAGE

| | | |
|-------|---|----|
| 1 | INTRODUCTION..... | 13 |
| 1.1 | Purpose and Scope | 13 |
| 1.2 | Galileo System Overview..... | 13 |
| 1.2.1 | Background | 13 |
| 1.2.2 | Frequency plan..... | 14 |
| 1.3 | System Requirements and Overview..... | 15 |
| 2 | SIGNAL PROCESSING ASPECTS | 18 |
| 2.1 | Sub-harmonic Sampling at RF..... | 18 |
| 2.2 | Mixer | 20 |
| 2.3 | Decimation..... | 20 |
| 2.4 | Requantization..... | 22 |
| 3 | BASELINE DESIGN OF THE DOWN CONVERTER..... | 23 |
| 3.1 | Baseline architecture | 23 |
| 3.2 | Mixer requirements and design..... | 27 |
| 3.3 | Filter requirements and design | 28 |
| 3.4 | Re-quantization requirements and design | 32 |
| 4 | MATLAB MODEL | 34 |
| 5 | HARDWARE DESCRIPTION..... | 35 |
| 5.1 | Introduction | 35 |
| 5.2 | Implementation | 37 |
| 5.2.1 | Mixers | 38 |
| 5.2.2 | Filters..... | 39 |
| 5.2.3 | Re-quantization | 40 |
| 6 | VERIFICATION | 41 |
| 6.1 | Methodology | 41 |
| 6.2 | Result..... | 41 |
| 7 | PERFORMANCE ANALYSIS..... | 42 |
| 7.1 | Noise error in ADC..... | 42 |
| 7.2 | ADC Non-linearity modeling..... | 43 |
| 7.2.1 | Introduction..... | 43 |
| 7.2.2 | Test method..... | 43 |
| 7.2.3 | Lookup Table modeling and analysis | 44 |
| 7.2.4 | ADC non-linearity modeling with LUT..... | 47 |
| 7.2.5 | ADC non-linearity modeling including spuri from ADC | 49 |
| 7.2.6 | Results and conclusion..... | 52 |
| 7.3 | ADC frequency response modeling | 52 |
| 7.3.1 | Introduction..... | 52 |
| 7.3.2 | Test method and setup..... | 53 |
| 7.3.3 | Analysis and result | 53 |
| 7.3.4 | Conclusion..... | 58 |
| 7.4 | SNR analysis for Down converter | 59 |
| 7.5 | FPGA Budgets..... | 60 |
| 8 | CONCLUSION | 61 |
| 9 | List of References | 62 |

(This page is intentionally left blank)

1 INTRODUCTION

1.1 Purpose and Scope

This report is based on the work performed in the master thesis work "Design of a Down Converter for a Galileo Receiver". The thesis work has designed a receiver front-end down converter for the new European navigation system, Galileo. The development platform Neptune-V5 VXS from Tekmicro has been used, which offers a high-speed RF sampling analogue-to-digital converter together with high-speed Xilinx Virtex-5 FPGAs. The task of this thesis work includes:

- defining a digital down converter architecture utilizing direct-RF sampling
- modeling the FPGA design in MATLAB
- implementing the design in an FPGA using VHDL design language and targeting an ASIC implementation considering size/power constraints
- assess the performance of key parameters through laboratory tests supported by analysis and/or system modeling

1.2 Galileo System Overview

1.2.1 Background

Named after the Italian astronomer Galileo Galilei, Galileo is a Global Navigation Satellite System being built by the European Union (EU) and European Space Agency (ESA). The system will become operational by the end of 2013.

The first Galileo spacecraft called Galileo In-Orbit Validation Element (GIOVE-A) was launched on 28 December, 2005 and a second spacecraft GIOVE-B was sent into space on 27 April 2008. The job of the spacecrafts was to ensure that the in-orbit technologies needed to run the Galileo system meets the frequency-filing allocation and reservation requirements from the International Telecommunication Union (ITU) [1].

Galileo is intended to be a civil system and will be run by a private consortium. Although GPS is nowadays used widely on civilian applications, the US military operator gives no guarantee to maintain an uninterrupted service. Therefore it is important for EU to build an independent system other than GPS.

Satellite navigation systems are mostly known to people from its in-car application. In addition to the open access navigation service, Galileo also offers four more services, i.e. commercial navigation, safety of life navigation, public regulated navigation and search and rescue service [1].

All these services require the Galileo system to offer great accuracy and great penetration. Each service should have an error less than a meter and it needs to serve well in urban centers, inside buildings, and under trees. Moreover, this system must also be able to tell the users if there are major errors that could compromise performance.

1.2.2 Frequency plan

The Galileo system occupies the frequency bands E1, E5a, E5b and E6 where E1, also known as L1, is a shared frequency band with the GPS system. E5a and E5b are two adjacent frequency bands. The combination of E5a and E5b is known as E5 and it fully covers the GPS bands L2 and L5 on the spectrum. The frequency of $f_0=1.023$ MHz is designed as the fundamental frequency of the Galileo system. All frequencies utilized in the system are generated from this fundamental frequency. A picture of the frequency allocation for both GPS and the Galileo system is shown in *Figure 1.1*.

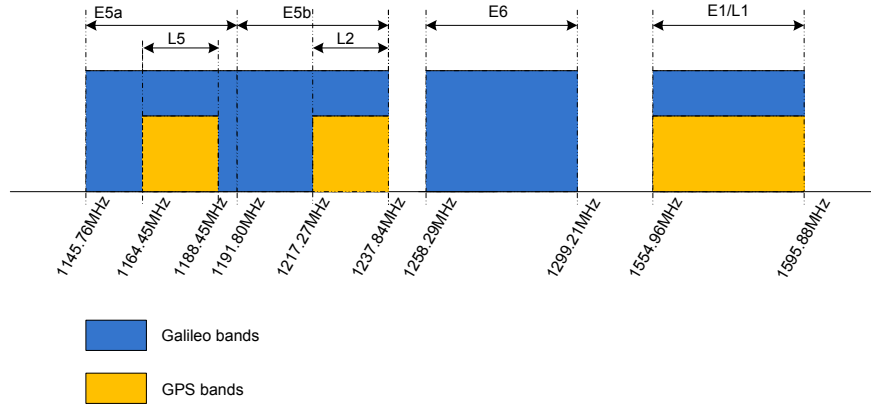


Figure 1.1: Illustration of Galileo & GPS system frequency allocation

The proposed receiver is designed aiming at processing signals on frequency bands E1, E5 and E6. This frequency scheme allows for receivers capable of handling both Galileo and GPS signals.

A summary of both the Galileo and the GPS system frequency plan, including the center frequency of each band, band width and lower and upper boundary for each band is listed in *Table 1.1*.

Table 1.1 Galileo system frequency plan

| System | Band | Center frequency f_c [MHz] | Band width BW [MHz] | Low boundary f_L [MHz] | Upper boundary f_H [MHz] |
|-------------|-------|------------------------------|--------------------------|-------------------------------|------------------------------|
| GPS/Galileo | L1/E1 | 1575.420 (1545 f_0) | 40.920 (40 f_0) | 1554.960 (1520 f_0) | 1595.880 (1560 f_0) |
| | GPS | 1227.600 (1200 f_0) | 20.460 (20 f_0) | 1217.370 (1190 f_0) | 1237.830 (1210 f_0) |
| GPS | L5 | 1176.450 (1150 f_0) | 24.000 (23.46 f_0) | 1164.450 (1138.270 f_0) | 1188.450 (1161.73 f_0) |
| Galileo | E5a | 1168.778 (1142.5 f_0) | 46.035 (45 f_0) | 1145.760 (1120 f_0) | 1191.795 (1165 f_0) |
| Galileo | E5b | 1214.812 (1187.5 f_0) | 46.035 (45 f_0) | 1191.795 (1165 f_0) | 1237.835 (1210 f_0) |
| Galileo | E6 | 1278.750 (1250 f_0) | 40.920 (40 f_0) | 1258.290 (1230 f_0) | 1299.21 (1270 f_0) |
| Galileo | f_0 | 1.023 | - | - | - |

1.3 System Requirements and Overview

The requirements defined at the down converter baseband outputs are:

- There shall be three outputs, each covering one Galileo signal (E5, E6 or E1).
- The output signals from the down converter shall be sampled at $120f_0$ (122.76 MHz).
- The output carrier to noise density ratio, C/N_0 , of the digital representation of any of the signal bands (E5, E6 and E1) shall be better than 117 dBHz.
- The amplitude variation versus frequency shall not exceed 3 dB peak-to-peak (*)
- The deviation from linear phase versus frequency shall not exceed 0.3 degrees peak-to-peak (*)
- Hardware implementation should operate with four samples in parallel at the input

(*) covers from ADC input to the output of the down converter

Limited by the FPGAs memory size and speed, the system should operate at a moderate rate which is not only determined by the sampling rate and the decimation factor of each stage but will also be affected by the filter length and its implementation cost. However, high sampling rate and large decimation factor requires a sharp filter with increased cost and complexity whereas a small decimation factor demands faster processing speed of the system which limits the performance of the output signal by the word length for each sample.

The highest sampling rate achievable by the onboard ADC for the Neptune-V5 VXS platform is 2.2GHz. This is a limitation which restricts the sampling rate of the down converter. Furthermore, a system sampling rate which is an integer multiple of $120f_0$, a frequency used to generate the navigation signal in Galileo system, is preferable but not obligatory.

It is desired for the hardware to operate with four samples in parallel at a rate of $f_s/4$. This is because if the hardware is moved to an ASIC it can interface directly with a 1:4 DEMUX which is described in section 5.1. This also means that the VHDL code to be developed will be designed with a future transfer to ASIC in mind to make the conversion easily.

A C/N_0 no worse than 117dB indicates that the SNR requirements for E1 and E6 signals is 41dB and for E5 signal is about 37dB in general. Here the SNR is calculated as

$$SNR_{[dB]} = C / N_{0[dB]} - 10 \log_{10}(BW) \quad (1.3.1)$$

where BW is the signal bandwidth.

Moreover, the requirement for a C/N_0 better than 117dB also constrains the total performance of the stopband attenuation of the anti-aliasing filter. To obtain the filter requirement, first assuming the signal power in each frequency band is equally distributed and has the same signal power,

$$S_{E1} = S_{E5} = S_{E6} \quad (1.3.2)$$

where S_E represents signal power on each frequency band.

Taking the E1 signal as an example; before downsampling, the SNR for E1 signal is calculated as,

$$SNR = \frac{S_{E1}}{N_{E1}} \quad (1.3.3)$$

where N_E represents the noise power on E1 band and can be derived from *Equation (1.3.3)*,

$$N_{E1} = SNR \cdot S_{E1} = S_{E1} \cdot 10^{-4}_{[watts]} \quad (1.3.4)$$

Assuming that the signals at the E5 and the E6 bands are completely fold back and become noise to E1 signal. Though for a real case only part of E5 and E6 signals fold back and affect E1 signal, this assumption is still accepted for the reason that it gives the worst case performance and allows for an easier analysis. Therefore, the SNR after downsampling is,

$$SNR = \frac{S_{E1}}{N_{E1} + (S_{E5} + S_{E6})G_S} \quad (1.3.5)$$

where G_S is the stopband attenuation of the anti-aliasing filter before downsampling.

By experience, a SNR degradation less than 0.2 dB is an acceptable level and is used in this analysis. In other words, the added new noise should be less than 4.7% of the original noise, that is to say,

$$\frac{(S_{E5} + S_{E6})G_S}{N_{E1}} \leq 4.7\% \quad (1.3.6)$$

From *Equation (1.3.2) and (1.3.6)*, the stopband attenuation G_S is calculated as

$$G_S \leq \frac{4.7\% \cdot N_{E1}}{S_{E5} + S_{E6}} \approx -56dB \quad (1.3.7)$$

Thus, in order to prevent the folded band from destroying the desired signal bands after downsampling, an anti-aliasing filter with stopband attenuation better than 56dB should be used.

To achieve the desired SNR, the filter requirements and linear phase deviation requires a low-noise and high-linearity system. These requirements are mainly limited by the ADC. This project is not focusing on the ADC design and therefore no effort is made to improve the performance of the ADC used. However the limitations of the ADC are analyzed in advance and compensated for at later stages in the project. Besides the inherent limitation from the ADC, the filter and re-quantization in the digital domain should also be carefully designed. The filter response will directly affect the in-band and out of band performance of the processed signal. The re-quantization steps will introduce more quantization noise other than the noise from ADC. It is always a compromise between performance and costs. How to design the filter coefficients and choose the re-quantized bits will be discussed in the following chapters.

For both verification and development purpose it is desirable to achieve a final setup and data paths as shown in *Figure 1.2*. This allows the output of the hardware implementation to be downloaded to the computer while still maintaining the ability to also download the RAW unprocessed data samples. By having access to both data MATLAB can be used to analyze the data or to perform further processing which is very useful.

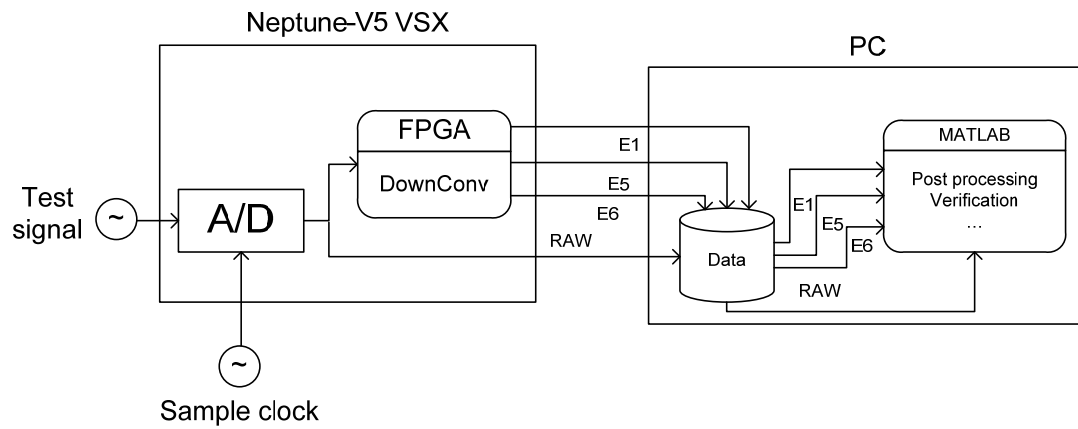


Figure 1.2: Schematic overview of complete setup

2 SIGNAL PROCESSING ASPECTS

2.1 Sub-harmonic Sampling at RF

The original Nyquist-Shannon sampling theorem states that a signal can be perfectly reconstructed from its sampled version only if the sampling frequency f_s is no less than twice of the highest frequency components (f_H) of that signal ($f_s \geq 2f_H$) [2].

In this thesis work, the input band of the receiver is approximately from 1GHz (lower edge of E5) to 1.7GHz (upper edge of E1). Applying the Shannon sampling theorem means the sampling rate of the ADC should be at least 3.4GHz. However the highest sampling rate for the onboard ADC is only 2.2GHz. Despite some inaccuracy such as non-linearity and temperature instability of analog components in the ADC, using 3.4GHz or even higher frequencies as the sampling frequency is power consuming and the ADC itself would be very expensive. An improved way to sample the band-pass signal is to use a technique called sub-harmonic sampling or undersampling.

According to the Fourier transform, sampling will generate harmonics of the spectrum at integer multiples of sampling frequency. Therefore the only criterion to avoid aliasing is to make sure there is no spectrum overlapping after sampling. In order to avoid aliasing, if the bandwidth of the signal is W and there are no frequency components above f_H , then one of the criterions the sampling frequency f_s should satisfy is $f_s \geq 2W$. Also, the band should not be placed over the border of two Nyquist zones. It is not necessary for f_s to be larger than twice the value of f_H ($f_s \geq 2f_H$). However, only keeping the sampling frequency larger than twice of the bandwidth cannot always avoid aliasing. More criterion need to be taken into consideration for finding f_s .

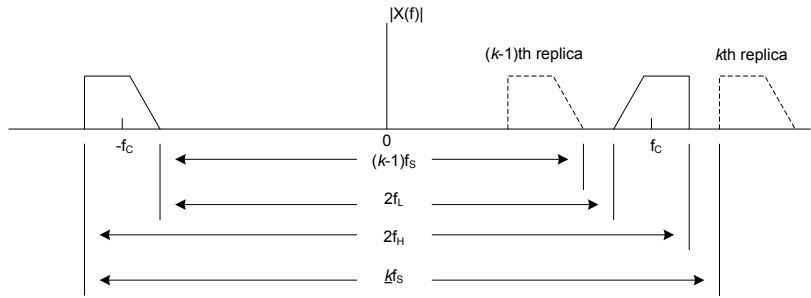


Figure 2.1: Illustration of sampling arbitrary bandpass signal

From Figure 2.1 one can see that it is possible to avoid aliasing if f_s satisfy the condition [3]:

$$\begin{aligned} 2f_H &\leq kf_s \\ (k-1)f_s &\leq 2f_L \end{aligned} \quad (2.1.1)$$

where k is an integer. f_H and f_L denotes the high and low band limits, respectively. The condition is a system of two inequalities with two unknown variables, k and f_s . But it can be easily solved and the constraints for finding f_s become:

$$\begin{aligned} \frac{2f_H}{k} &\leq f_s \leq \frac{2f_L}{k-1} \\ 1 &\leq k_{\max} \leq \left\lfloor \frac{f_H}{W} \right\rfloor \end{aligned} \quad (2.1.2)$$

The largest value of k which satisfies the second condition leads to the lowest f_s .

The input signal to the receiver is a combination of signals in the three frequency bands E1, E5 and E6. Each of them has a bandwidth of 50 MHz, 90 MHz and 50 MHz, respectively. To fulfill the sampling theorem, f_s should satisfy the condition in *Equation (2.1.2)*. A single sampling frequency should be used in the system rather than three different rates (one for each band) for the reason that only one ADC is available on the developing platform. After sampling, the utilized band should be in a reasonable frequency range and without spectrum overlapping. The three signal bands should be separated enough in order to reduce the requirements of implementing an anti-aliasing filter before downsampling at later stages.

For the work represented here, a MATLAB script is written to choose the sampling frequency. It returns a list of sampling frequencies which satisfies the undersampling condition *Equation (2.1.2)*. The code selects utilized frequency ranges for different sampling rates and then compares the margin of the three spectrums at the selected frequency ranges to each other as well as to zero and the Nyquist frequency zone. After comparison, two frequencies at $720f_0$ and $960f_0$ are selected. Both frequencies are integer multiples of $120f_0$. Moreover, both frequencies utilized sub-harmonic sampling without spectrum overlapping and have similar spectrum characteristics at selected reasonable frequency range. The sampling frequency $720f_0$ is preferable for the final implementation because it allows ADC to sample at a lower sampling rate and prominently reduces the hardware requirement compared to the sampling frequency $960f_0$.

A real signal has its spectrum symmetric to zero. The negative frequency spectrum of the real signal is a mirrored image of the positive frequency spectrum. Therefore, the harmonic will be mirrored if the signal appears at even order of Nyquist zone (*Figure 2.2a* and *Figure 2.2b*). In contrast, the signal placed in an odd Nyquist zone will not be mirrored (*Figure 2.2c* and *Figure 2.2d*).

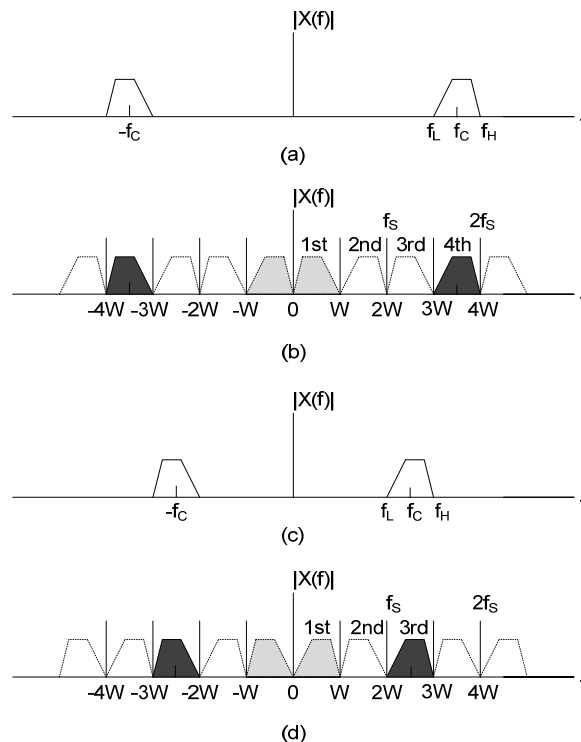


Figure 2.2: Illustration of bandpass signal sampling

In the work presented here, after sub-harmonic sampling the E1 band appears at $1540f_0 - 2 \cdot 720f_0 = 100f_0$ and since the E1 band is in the 5th Nyquist zone, the spectrum is not mirrored. The E5 band appears at $1165f_0 - 2 \cdot 720f_0 = -275f_0$ and the E6 band appears at $1250f_0 - 2 \cdot 720f_0 = -190f_0$. Both E5 and E6 are in the 4th Nyquist zone so mirrored spectrums are utilized.

2.2 Mixer

For a system that receives a signal at RF, a frequency mixer is usually used to bring down the input signal from RF to baseband. A frequency mixer accepts two signals, one input signal and one mixing signal that is generated by a local numerically controlled oscillator (NCO). The mixer then multiplies these signals together and output signals at the sum and the difference of the frequencies.

Suppose the representation for a real signal is $s=A_1\sin(2\pi f_1t)$ where f_1 is the carrier frequency of the signal. To bring down the frequency, the local oscillator generates a mixing signal $s_0(t)=A_2\sin(2\pi f_2t)$, where f_2 is generated by the NCO. Then the mixer output will have the form of,

$$\begin{aligned} s'(t) &= A_1 \sin(2\pi f_1 t) \cdot A_2 \sin(2\pi f_2 t) \\ &= \frac{A_1 - A_2}{2} [\cos(2\pi(f_1 - f_2)t) + \cos(2\pi(f_1 + f_2)t)] \end{aligned} \quad (2.2.1)$$

It is well known that the Fourier Transform of a cosine function is

$$\cos(2\pi f' t) \xrightarrow{FT} \delta(f - f') + \delta(f + f') \quad (2.2.2)$$

Therefore taking the Fourier Transform for (2.2.1) at both sides gives

$$\begin{aligned} S'(f) &= \frac{A_1 - A_2}{2} [\delta(f - (f_1 - f_2)) + \delta(f + (f_1 - f_2))] \\ &\quad + \delta(f - (f_1 + f_2)) + \delta(f + (f_1 + f_2))] \end{aligned} \quad (2.2.3)$$

Therefore the result appears at $|f_1+f_2|$ and $|f_1-f_2|$ as shown in *Figure 2.3*. A low pass filter will also be required to filter away the unwanted spectrum to avoid aliasing in later stages.

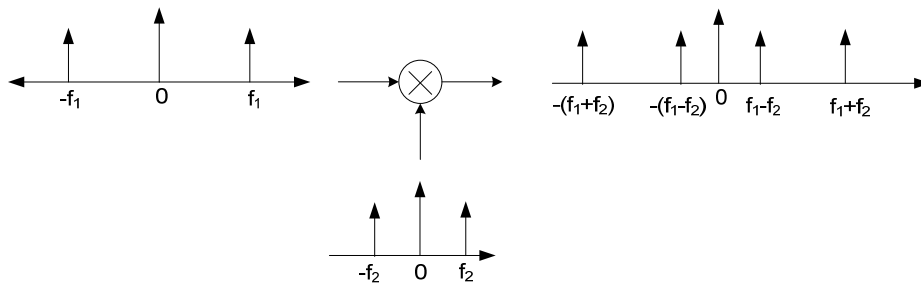


Figure 2.3: Illustration of mixing with real signal

2.3 Decimation

In this proposed down-converter architecture, the technique of decimation is taken into consideration. Decimation is a technique used to reduce the number of samples in a system. It includes two steps: a low pass anti-aliasing filter and downsampling. Using $f_s=720f_0$ for the entire system will be a difficult and inefficient solution but using a lower sampling frequency will reduce the complexity and consumes less power for the hardware. It turns out to be effective for the system to reduce the sampling rate before further processing of the signals.

Downsampling is a process used to reduce the sampling rate. The downsampling factor (denoted by D in this thesis) is usually an integer for a digital system. The process, shown in *Figure 2.4*, picks up every D^{th} samples and discards the rest of samples in that signal. Assume $p(n)$ is a periodic impulse response with period D (*Figure 2.4b*) and $v(n)$ is the incoming signal (*Figure 2.4a*), a multiplication of the two signals ($v'(n)=v(n)p(n)$) results in a sequence of signals showed in *Figure 2.4c*. $v'(n)=0$ except at multiples of D . Finally by removing zeros in $v'(n)$, the result becomes $y(n)=v'(nD)=v(nD)$, (as shown in *Figure 2.4d*). Consequently, the number of samples is reduced to $1/D$ of the original amount and the sampling rate is brought down from f_s to f_s/D [3].

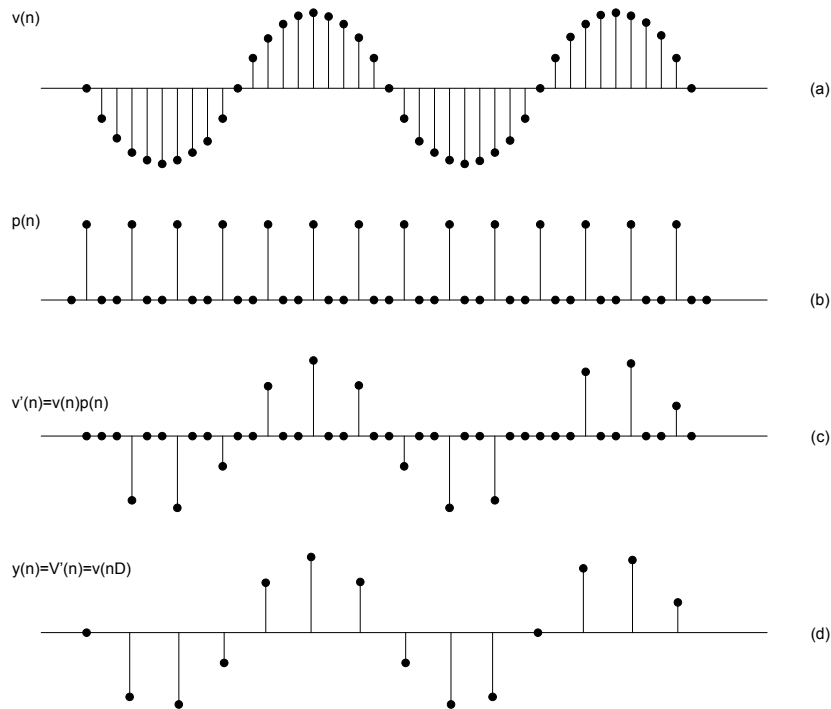


Figure 2.4: Illustration of downsampling by a factor D

As downsampling reduces the sampling frequency, it is important to make sure that the Nyquist sampling theorem is still satisfied otherwise aliasing will occur. For that reason, a low-pass filter as an anti-aliasing filter is utilized before the downsampling.

Designing a finite impulse response (FIR) filter involves choosing stopband attenuation, in band ripple and cut off frequencies (how to choose these are explained in chapter 3.3). These criterions will decide the filter coefficients which represents the impulse response of the proposed filter design. These coefficients, in linear convolution with the input sequence will result in the desired output. The linear convolution process can be represented as:

$$y[n] = x[n] * h[n] \quad (2.3.1)$$

here, $y[n]$ signifies the output of the filter and $x[n]$ is the input to the filter and the impulse response is called $h[n]$.

The impulse response of the filter is given by $f[n]$ and the operator ‘*’ denotes the convolution operation. For a finite impulse response (FIR) filter, the above representation changes to:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{L-1} h[k] x[n-k] \quad (2.3.2)$$

where L is the order of the filter.

2.4 Requantization

Compared to floating point arithmetic, fixed-point arithmetic makes it possible for the system to have a higher speed and lower power consumption. Considering these advantages of fixed-point arithmetic, it was chosen for the proposed architecture. In order to keep the growth of word lengths within boundaries, rounding and truncation need to be applied to the number of bits which are used to represent the signal while maintaining the correct information of each sample. This procedure is called re-quantization.

In the work presented here, both the frequency mixer and the filter stage will introduce bit growth during the multiplication operation. Assuming that the frequency mixer multiplies the input signal with an n -bits complex mixing signal and the fixed-point representation of the filter coefficients is m bits, the output signal will be grown to $10+n+m$ bits with a 10-bit ADC applying the input signal. A large growth in word size without re-quantization will not only slow down the operation speed but might also go beyond the limitation of the memory size at a high sampling rate. However using too few bits after re-quantization may cause the signal to saturate to its minimum or maximum value if the output signal is stronger than the full range of selected bits. Therefore the process of truncation and rounding needs to be carefully designed. The hardware implementation of the re-quantization will be explained in further details in section 5.3.3. To optimize the system for best efficiency, re-quantization is applied after each frequency mixer and filter stages. The performance after each re-quantization stage will be analyzed in detail in section 6.2.

3 BASELINE DESIGN OF THE DOWN CONVERTER

3.1 Baseline architecture

The proposed down converter should be able to minimize the hardware cost and implementation complexity due to the high sampling rate of the ADC. For this reason, a cascade stage architecture, shown in *Figure 3.1*, seems to be an efficient solution. Each down-converter stage of the proposed architecture includes a frequency mixer, an anti-aliasing filter, downsampling and re-quantization stages. The purpose of this architecture is to postpone as much as possible of the filtering steps to later stages, operating at lower sampling rates where they can be implemented more efficiently.

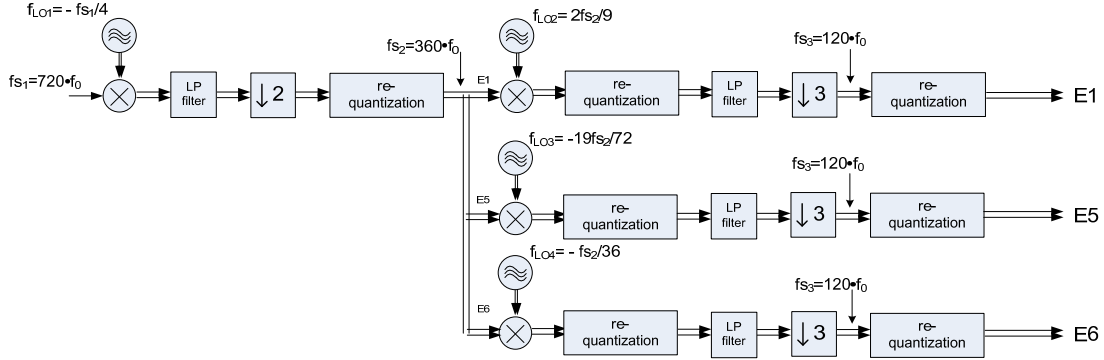


Figure 3.1: Proposed down converter architecture

As mentioned in section 2.1, a sampling frequency of $720f_0$ is chosen. *Table 3.1* lists the new center frequency allocations for the bands E1, E5 and E6 after sub-harmonic sampling. For E5 and E6, mirrored images at positive frequencies are used to optimize system implementation. *Figure 3.2a* shows the symbolic spectrum after sampling which doesn't contain any information of the processed signal except the frequency allocation of each band after operation.

Table 3.1: Frequency allocation after RF-sampling

| Signal | f_c/f_0 | f_{s1}/f_0 | f_c'/f_0 | comments |
|--------|-----------|--------------|------------|----------------------------|
| E1 | 1540 | 720 | 100 | - |
| E5 | 1165 | 720 | 275 | Mirrored spectrum utilized |
| E6 | 1250 | 720 | 190 | Mirrored spectrum utilized |

After the ADC, a frequency mixer is used to mix the sampled signal with a complex signal at frequency $f_{LO1}=f_s/4=180f_0$ to bring down each frequency band to a lower frequency zone. This step also converts the input real signal to a complex signal. The spectrum after this stage is shown in *Figure 3.2b*.

The 1st stage decimator downsamples each signal by a factor of 2 to reduce the sampling rate. The new sampling frequency is $f_{s2}=360f_0$ and the new Nyquist zone for the processed complex signal becomes $-180f_0$ to $180f_0$. After downsampling the new sampling frequency should still satisfy the condition described in section 2.2. A low pass filter is used before the downsampling to avoid aliasing. The spectrum after the 1st stage low-pass filter and downsampling are shown in *Figure 3.2c* and *Figure 3.2d* respectively.

The frequency of the 1st stage complex mixer signal is $f_s/4$ which allows the hardware to be implemented very efficiently due to mixer points only occur at 0, 90, 180 and 270 degrees resulting in simple sine/cosine values. Because of this, it will not introduce any bits growth into the word length. Only the low pass filter brings bits growth into the system, so re-quantization is carried out after decimation in the 1st stage.

The 2nd cascade stage of the down converter has an architecture similar to the 1st stage except that it processes the signal in 3 different branches, each branch focus on one of the three bands, E1, E5 and E6.

For the E1 branch, the frequency mixer brings down the processed signal to base band by mixing it with a complex signal at $2f_{S2}/9=80f_0$. For the E5 and E6 branches, a complex mixing signal at $-19f_{S2}/72=-95f_0$ and $-f_{S2}/36=-10f_0$ is used respectively. Consequently, the 2nd stage frequency mixer brings down both processed signal to baseband as expected.

As is reflected in *Figure 3.2e* to *Figure 3.2g*, the spectrum of each branch after the 2nd stage is at baseband as expected but the signal is still sampled at a high frequency. Therefore, a 2nd decimation stage is applied to further reduce the sampling rate by a factor of 3 and low-pass filters are used as anti-aliasing filters before downsampling in each branch. The sampling frequency at the output connected to the next processing unit is $f_{S3}=f_{S2}/3=120f_0$. The symbolic spectrums after filtering and down-sampling are shown in *Figure 3.2h* to *Figure 3.2m*.

In the 2nd cascade stage, the mixed complex signal and filter coefficients need to be quantized to fixed-point notation when implementing the hardware. The conversion from floating-point to fixed-point representation of the complex mixing signal and the anti-aliasing filter in the 2nd stage, introduces bit growth to the samples. In order to minimize the word length for each sample while maintaining a satisfactory performance at the output, the re-quantization step is carefully designed and applied twice both after the frequency mixer and after the anti-aliasing filter in this stage.

The input rate for the down converter is 4 samples per clock cycle according to the property of the ADC which outputs 4 samples per system clock cycle. Therefore the 1st stage works on 4 samples per cycle and 2nd stage works on 2 samples on each branch in parallel. The final output of the down converter has 2 samples per 3 clock cycles due to the decimation by 3 at 2nd stage.

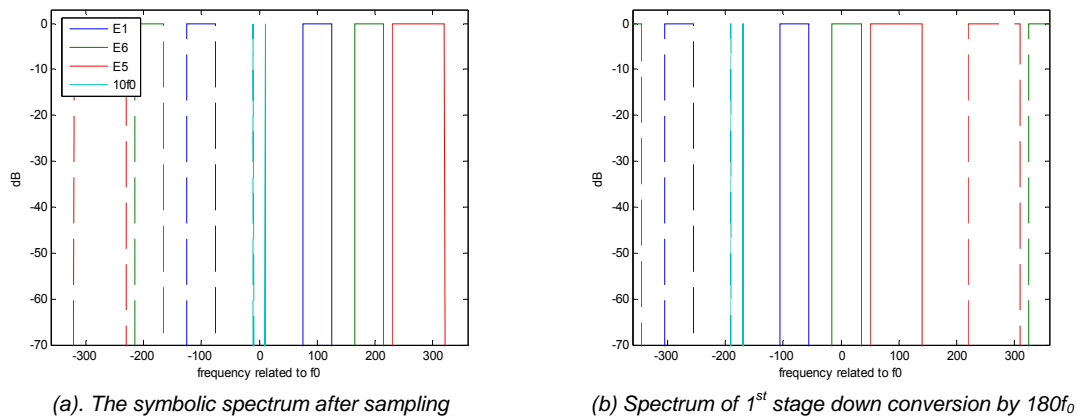
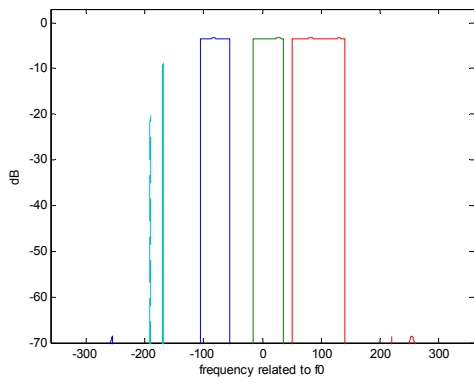
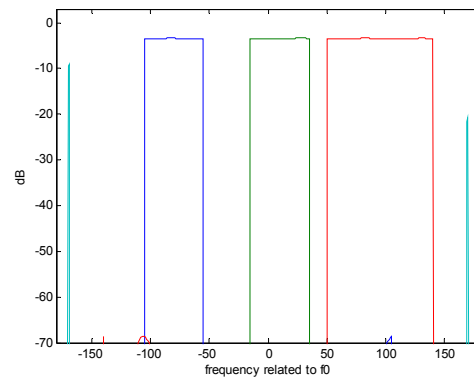


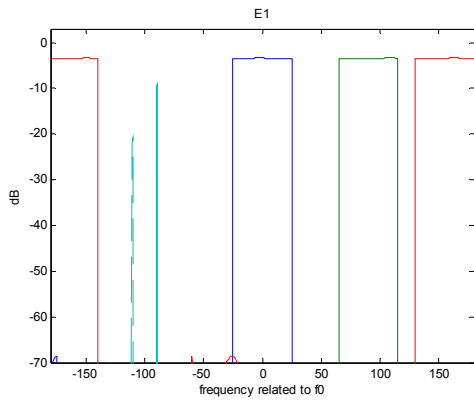
Figure 3.2: Spectrum of proposed down converter



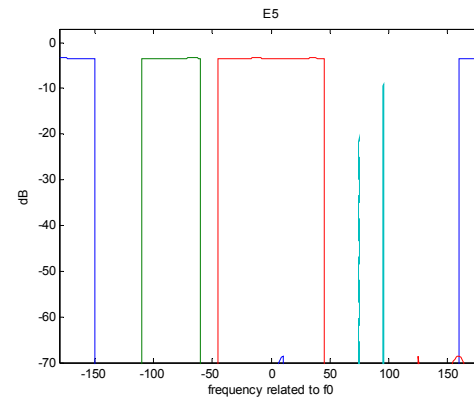
(c) 1st stage, spectrum after low-pass filter



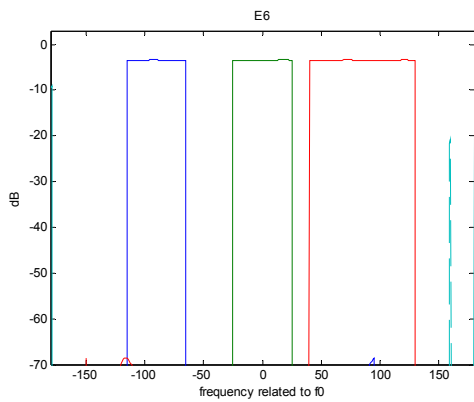
(d) 1st stage, spectrum after down-sampling by 2



(e) E1 branch 2nd stage, spectrum after frequency mixer

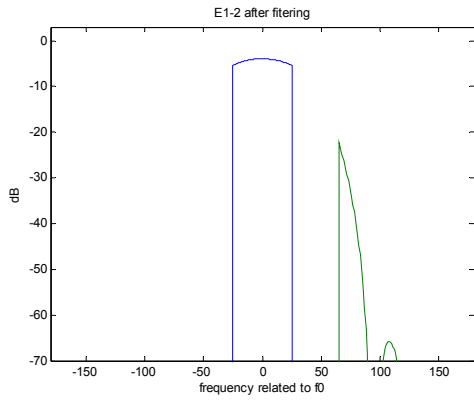


(f) E5 branch 2nd stage, spectrum after frequency mixer

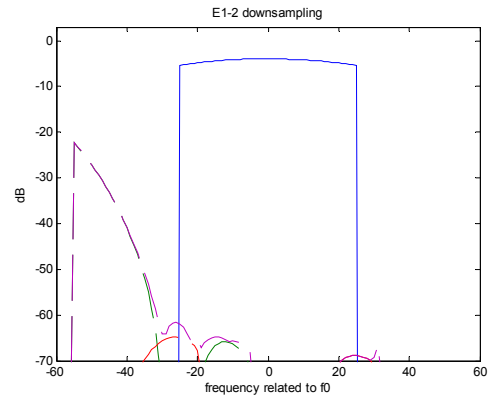


(g) E6 branch 2nd stage, spectrum after frequency mixer

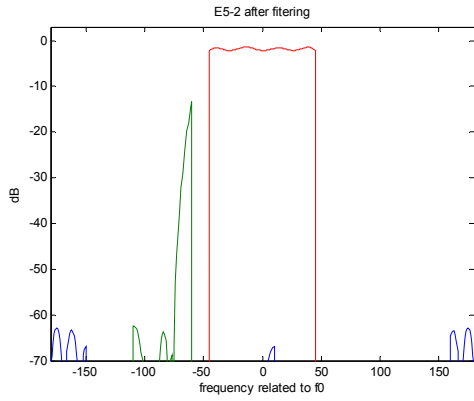
Figure 3.2: Spectrum of proposed down converter



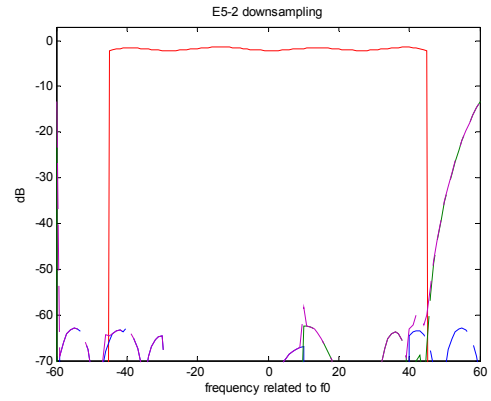
(h) E1 branch 2nd stage, spectrum after filtering



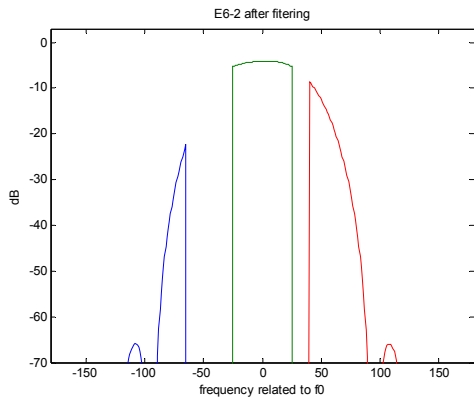
(i) E1 branch 2nd stage, spectrum at output



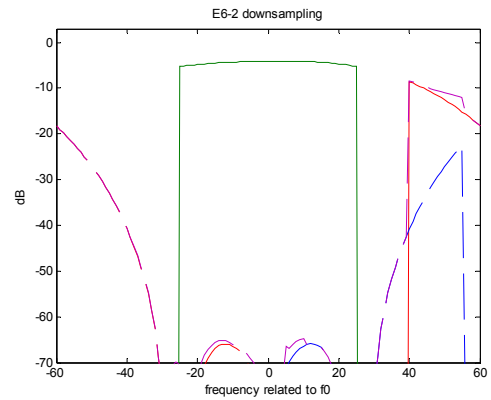
(j) E5 branch 2nd stage, spectrum after filtering



(k) E5 branch 2nd stage, spectrum at output



(l) E6 branch 2nd stage, spectrum after filtering



(m) E6 branch 2nd stage, spectrum at output

The mapping scheme used in Figure 3.2 is:

E1.....blue

E5.....red

E6.....green

10f₀.....cyan

Total noise power purple

Desired signal.....solid

Redundant signal.....dashed

Figure 3.2: Spectrum of proposed down converter

3.2 Mixer requirements and design

The denominator of the ratio f_{LO}/f_S determines the size of the mixer. To implement the mixer in hardware, it needs to be quantized to fixed-point notation and the number of bits used to represent the mixer points must be carefully chosen. Using more quantization bits gives a lower quantization noise but it is power consuming. However an efficient implementation introduces more quantization noise to the signal. It is a trade of between hardware complexity and quantization error. The mixer with the frequency $f_{LO}=f_S/4$ allows for a very efficient implementation since there are only 4 points used in the mixer and none of them needs to be quantized to fixed-point notation. Therefore no re-quantization steps are needed after the 1st stage mixer.

After the 1st mixer stage, a frequency offset remains on the down-converted signal. Therefore mixers with frequencies $2f_S/9$, $-19f_S/72$ and $-f_S/36$ are introduced for E1, E5 and E6, respectively. By the help of MATLAB, 9 bits are decided to quantize the mixer.

Figure 3.3 shows the spectrum of each mixer after quantization with 9 bits.

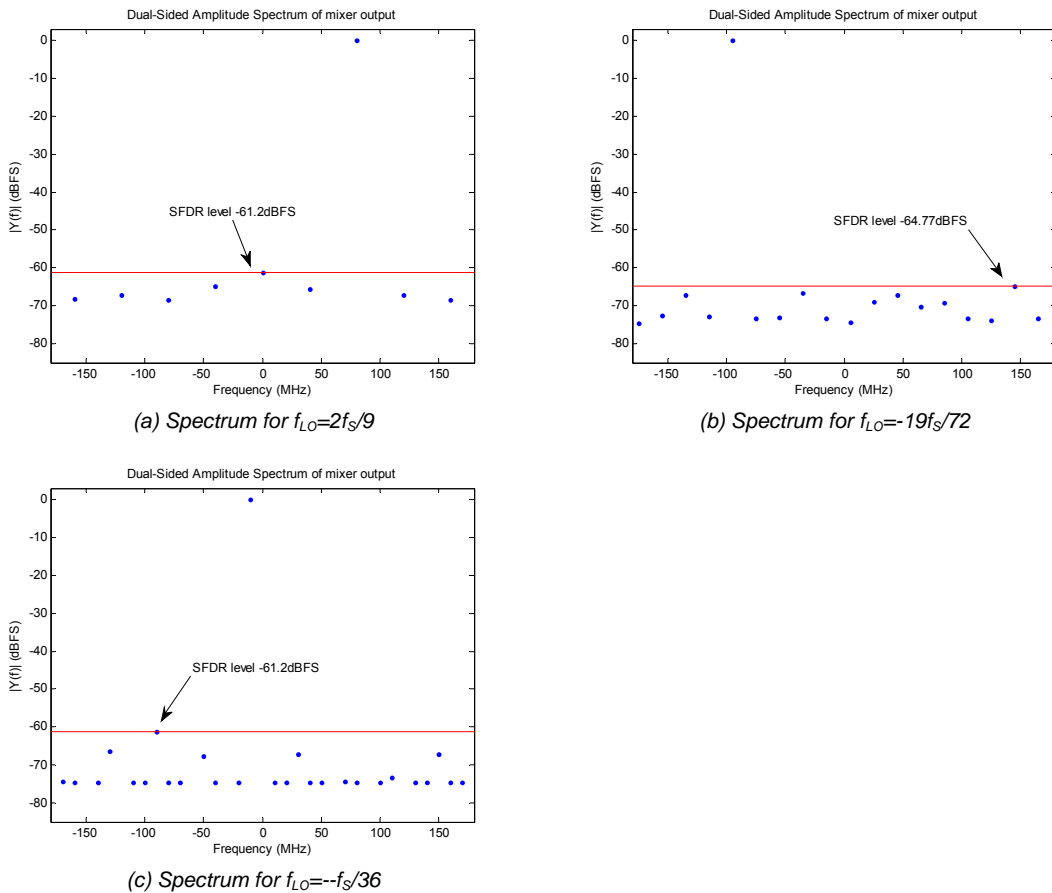


Figure 3.3: Quantized mixer performance

The SNR is 56.58 dB, 57.58 dB and 56.38 dB for mixers at E1, E5 and E6, respectively. Moreover, the intrinsic quantization noise floor Q_0 is calculated as

$$Q_0 = SNR \cdot BW \quad (3.2.1)$$

Table 3.2 summarizes the calculated parameters.

Table 3.2: Mixer performance

| | SFDR [dBFS] | SNR [dB] | Q_0 [dBc/Hz] |
|-------------------------------------|----------------|-------------|-------------------|
| E1 ($f_{LO}=2f_s/9$) | -61.2 | 56.58 | -142.24 |
| E5 ($f_{LO}=-19f_s/72$) | -64.77 | 57.58 | -143.24 |
| E6 ($f_{LO}=-f_s/36$) | -61.2 | 56.38 | -142.04 |

Table 3.3 lists the chosen mixer frequencies and the frequency offset Δf that remains on the down-converted signal after each stage. This Δf will be corrected in later stages where the processing can be implemented more efficiently.

Table 3.3: Local oscillator frequency

| 1st stage Mixer | Signal | f_c/f_0 | f_{LO1}/f_0 | f_{LO1}/f_{s1} | $\Delta f/f_0$ |
|---|---------------|-----------|---------------|------------------|----------------|
| | E1 | 100 | -180 | -1/4 | -80 |
| | E5 | 275 | -180 | -1/4 | 95 |
| | E6 | 190 | -180 | -1/4 | 10 |
| 2nd stage Mixer | Signal | f_c/f_0 | f_{LO2}/f_0 | f_{LO2}/f_{s2} | $\Delta f/f_0$ |
| | E1 | -80 | 80 | 2/9 | 0 |
| | E5 | 95 | -95 | -19/72 | 0 |
| | E6 | 10 | -10 | -1/36 | 0 |

3.3 Filter requirements and design

Calculating the coefficients of a digital filter involves considerable amount of computation. To be able to use an FIR filter in an FPGA, the coefficients need to be quantized. By experience, the quantization of the filter coefficients can sometimes put the filter out of specification. Moreover, the cost to implement the filter coefficients directly affects the speed and complexity of the proposed architecture. Taking all the aspects mentioned above into consideration, a MATLAB code is developed to assist in the design of the filter.

In order to avoid massive mathematical calculations, a more common industrial approach has been chosen to find the preferred digital filter. A MATLAB code has been developed to serve this propose. A general requirement for designing the filter, in this project, was that it should have a passband ripple less than 3dB and stopband attenuation larger than 56dB. The desired filter should also be able to filter away the frequencies outside of the Nyquist zone before downsampling to avoid aliasing. Given the basic criterions, on cutoff frequency and sampling frequency, the developed MATLAB code searches for all possibilities that fulfill the specified requirements. Then the one that has the smallest passband ripple, largest stop band attenuation and least number of adders will be selected. Only FIR filters with linear phase were considered in this case, therefore no phase distortion will be introduced. In this way, the best performance filter is found efficiently. Moreover, this method has the best flexibility. It can easily be modified to find digital FIR filters with different criterions in various situations.

When designing a filter, the implementation cost is usually inversely proportional to the transition bandwidth and passband ripple but it is directly proportional to the stopband attenuation. As the system requires a total in-band ripple no larger than 3 dB and out of band attenuation higher than 56 dB, an in-band ripple less than 2 dB and out of band attenuation higher than 60 dB have been used as the designing goal for each filter. However these criteria are flexible and can be adjusted around the general goal depending on the sampling frequency and the transition band of the filter. A lower sampling rate with a wider transition band allows for a cheaper implementation of the filter. Therefore it is feasible to raise the requirements for such a filter to archive better performance by trading off power dissipation and complexity, yet maintaining a cheap implementation in general.

After downsampling the signal by a factor of D , the size of the new Nyquist zone ($\pm f_s'/2$) changes to $1/D^{\text{th}}$ of the original ($\pm f_s/2$) and the signal between the new and old edges ($-f_s/2 \sim -f_s'/2$ and $f_s'/2 \sim f_s/2$) folds back to the new Nyquist zone. In order to avoid aliasing, the designed filter should be able to filter away signals between these regions. Moreover, as mentioned above, a wider transition band allows for a more efficient implementation. Taking all aspects into consideration, the passband frequency f_{pass} has been chosen as,

$$\begin{aligned} f_{\text{pass}} &= f_H \text{ for } f_H \geq |f_L| \\ f_{\text{pass}} &= |f_L| \text{ otherwise} \end{aligned} \quad (3.3.1)$$

where f_H and f_L denote upper and lower edge of the band, respectively.

Then with the help of f_{pass} , the stopband frequency f_{stop} is calculated as,

$$f_{\text{stop}} = f_s' - f_{\text{pass}} \quad (3.3.2)$$

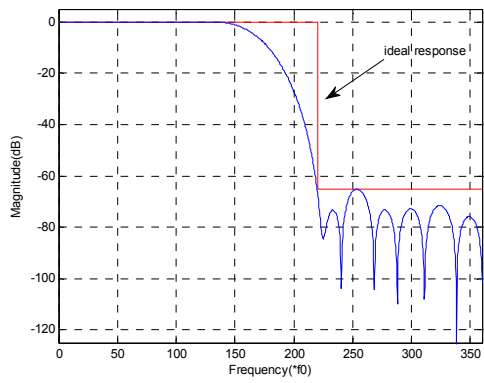
where f_s' is the new sampling rate after downsampling by factor D .

Table 3.4 lists the chosen cutoff frequency and the performance for each filter. Note that the 2nd stage filter for E1 and E6 has $f_{\text{stop}}=90f_0$ MHz instead of $95f_0$ MHz. This filter operates at a lower sampling rate, and by experience it is possible to trade off the complexity, within acceptable levels, for a better performance. With the help of the developed MATLAB code it is found that although the performance is better with $f_{\text{pass}}=90f_0$ MHz, there is no noticeable difference for complexity between the two frequencies.

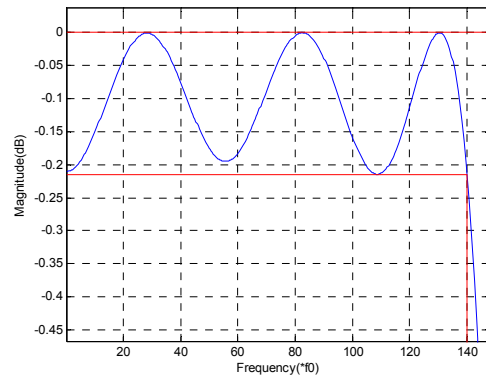
Table 3.4: Filter parameters

| | Cutoff frequency | | Passband ripple [dB] | Stopband attenuation [dB] | Filter coefficients | |
|-----------------------|--------------------------|--------------------------|----------------------|---------------------------|--|--|
| | $f_{\text{pass}} (*f_0)$ | $f_{\text{stop}} (*f_0)$ | | | | |
| 1 st stage | 140 | 220 | 0.21 | 65.03 | [-2, -12, -17, 7, 34, -1, -64, -18, 110, 61, -194, -178, 449, 1178, 1178, 449, -178, -194, 61, 110, -18, -64, -1, 34, 7, -17, -12, -2] | |
| 2 nd stage | E1/E6 | 25 | 90 | 1.27 | 60.58 | [-1, -4, -6, 1, 25, 64, 102, 118, 102, 64, 25, 1, -6, -4, -1] |
| | E5 | 45 | 75 | 0.67 | 60.21 | [1, -1, -9, -24, -34, -24, 13, 55, 59, -1, -97, -144, -51, 200, 515, 734, 734, 515, 200, -51, -144, -97, -1, 59, 55, 13, -24, -34, -24, -9, -1, 1] |

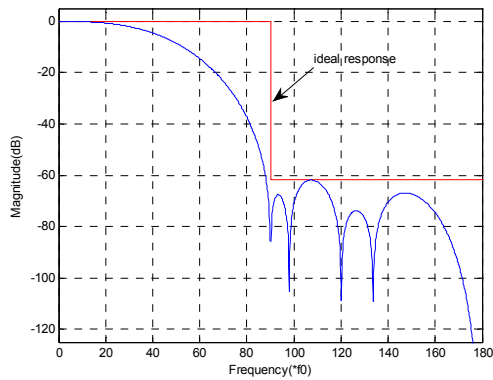
Figure 3.4 shows the frequency response of the designed filters. Since only linear phase filters are considered in this project no phase response curves are given in the figure.



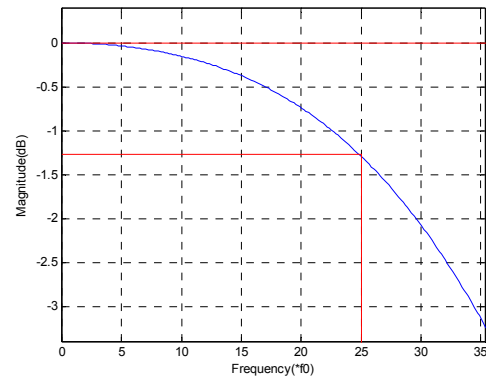
(a) 1st stage filter response, normalized gain, ref=68.86dB



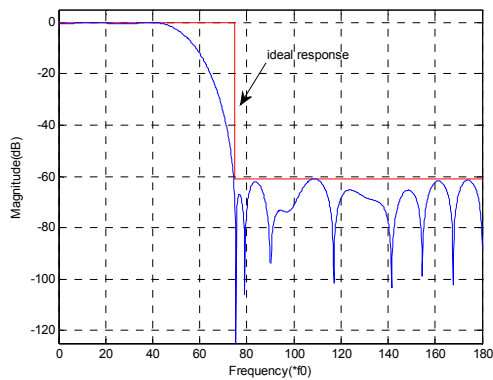
(b) Passband distortion



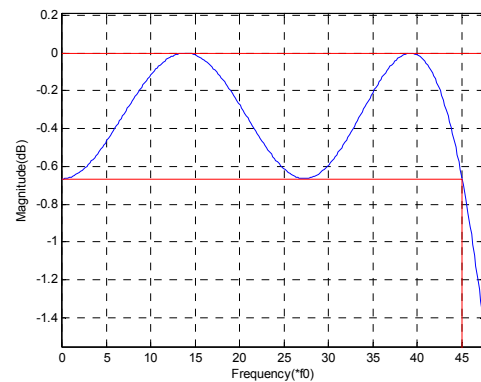
(c) 2nd stage filter for E1 and E6 branches, normalized gain, ref=53.62dB



(d) Passband distortion



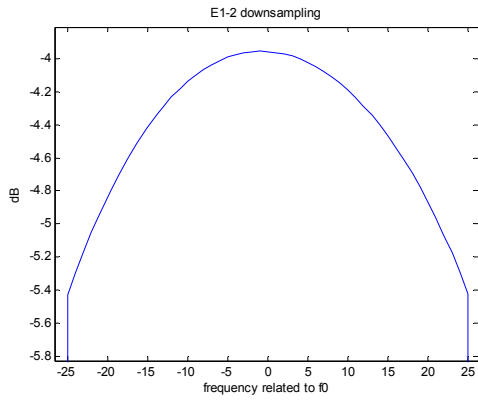
(e) 2nd stage E5 branch, normalized gain, ref=68.21dB



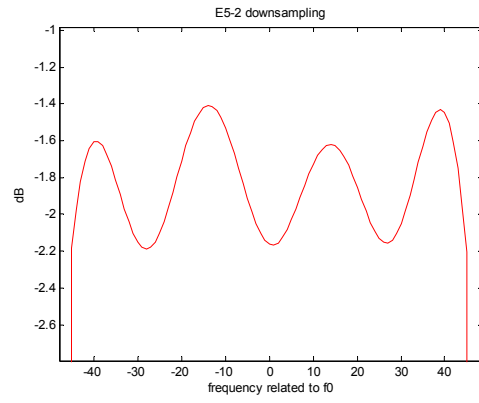
(f) Passband distortion

Figure 3.4: Filter amplitude responses

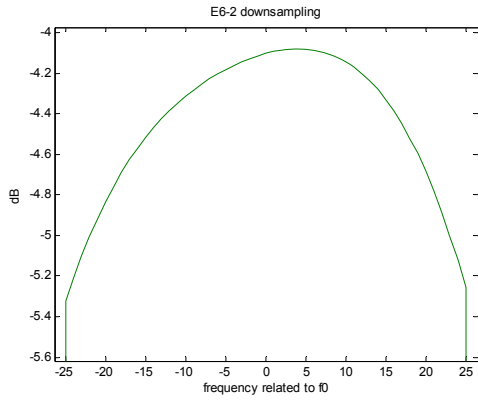
As shown by the simulation results in *Figure 3.5a* to *Figure 3.5c*, the total in-band ripple at the output of the down converter on each branch is 1.48 dB, 0.78 dB and 1.24 dB for E1, E5 and E6, respectively. One of the system requirements for the receiver is to assure in-band ripple no higher than 3 dB. The real case ripple may differ from this theoretical estimation. However the simulation results are far below 3dB and the conclusion can be made, so far, that this is an acceptable performance.



(a) E1, passband ripple at output



(b) E5 passband ripple at output



(c) E6, passband ripple at output

Figure 3.5: Overall filter in-band response (including both filters)

3.4 Re-quantization requirements and design

As mentioned in the sections above, both filtering and mixing stages will introduce bit growth in the system. Due to the limited speed of the hardware, the growth of word length must be kept within boundaries. Moreover, from the perspective of optimizing the hardware complexity and power consumption, rounding and truncation need to be applied after the multiplication, where word length has grown, while maintaining the correct information of each sample.

To illustrate how re-quantization is carried out in this project, the operation after the first stage filtering is taken as an example. The 1st stage filter introduces 13 bits into each sample and results in a 23 bits output after filtering. The first 2 bits are discarded because only 21 bits are used with a reasonable input signal levels. To further minimize the number of bits, 9 LSB in the sample are also cut away so that only 12 bits are left after the first re-quantization stage. However, if the signal is stronger than the full range of selected bits, -2047 to 2047 in this case, then the signal will be saturated to its minimum or maximum value after re-quantization. The re-quantization will use a symmetrical output to make the hardware implementation easier, that is why -2047 ($-2^{11}+1$) is used instead of -2048 (-2^{11}). To determine the reasonable number of bits used for re-quantization, signals are generated from the synthesizer with frequencies $1545f_0$, $1165f_0$ and $1250f_0$ to represent signal at the E1, E5 and E6 bands, respectively. Each input signal has a nominal level 136 mV (-11.3 dBFS), 124 mV (-12.1 dBFS) and 132 mV (-11.6 dBFS) respectively. However the hardware implementation will be designed to allow individual signals up to -5 dBFS to have some margin allowing for temporal fluctuations. This margin is approximately 6 dB = 1 bit. *Table 3.5* lists the chosen number of bits before and after re-quantization.

Table 3.5: allocation on each stage

| | | E1 | E5 | E6 |
|--------------------------|---|-----------|------------|-----------|
| | ADC input, individual (pp) [mV] | 136 | 124 | 132 |
| | ADC full scale input (pp) [mV] | | 500 | |
| | Input [bits] | | 10 | |
| 1 st stage | After filtering [bits] | | 23 | |
| | After re- quantization [bits] | | 12 (21:10) | |
| 2 nd stage | After 2 nd mixer [bits] | 21 | 21 | 21 |
| | After re- quantization [bits] | 13 (20:8) | 13 (20:8) | 13 (20:8) |
| | After filtering [bits] | 22 | 25 | 22 |
| | After re- quantization / final output [bits] | 14 (21:8) | 14 (23:10) | 14 (21:8) |

MATLAB simulations have been performed to deduce how to use the hardware bits efficiently. To do that the filter output is scaled with a constant to fully use the hardware bits in the worst case scenario which in this case was decided to -5 dBFS. In *Table 3.6* the filter scaling factors and worst and nominal case bit allocations are shown.

Table 3.6: Filter scaling factors and bit allocations

| | | E1 | E5 | E6 |
|-----------------------------|---|-----------|-----------|-----------|
| 1st stage | Filter scaling factor | | 1.5 | |
| | Bit allocation nominal case [bits] | | 11.71 | |
| | Bit allocation worst case [bits] | | 11.93 | |
| 2nd stage | Filter scaling factor | 0.875 | 0.75 | 0.875 |
| | Bit allocation nominal case [bits] | 12.85 | 12.79 | 12.78 |
| | Bit allocation worst case [bits] | 13.90 | 13.97 | 13.87 |

4 MATLAB MODEL

During the work process a MATLAB model has been developed that implements the proposed architecture in the time domain. This model exists in two versions where one is a floating point model and the other is a bit true model. The differences between these versions are shown below.

Floating point model

- Uses floating point precision sine/cosine values in mixers
- Does not requantize, keeps floating point through whole design

Bit true model

- Uses quantized (9 bit) precision sine/cosine values in mixers
- Requantizes to reduce hardware complexity

The MATLAB model is used for verification and to compute the total quantization error. Both versions have the same function header so they can be easily compared.

5 HARDWARE DESCRIPTION

5.1 Introduction

The hardware used in this project is a version of the Tekmicro Neptune-V5 VXS board equipped with three Virtex5 FPGAs, one 2.2 GSPS ADC (e2v AT84AS008), 1 GB DDR3 SDRAM/FPGA and gigabit Ethernet as a module in one of the SFP+ slots. A block diagram is shown in *Figure 5.1*. The empty block at the IOTOP FPGA is a slot where either a DAC or a second ADC can be mounted. Due to the high sampling rate capabilities of the ADC an ASIC DEMUX is placed between the ADC and the FPGA to ease the speed requirements on the FPGA. This DEMUX brings down the rate by a factor of 4 meaning that the FPGA receives 4 samples in parallel at a rate of a quarter of the sampling frequency. Further information about the Neptune-V5 VXS board can be found at <http://www.tekmicro.com/>. [4]

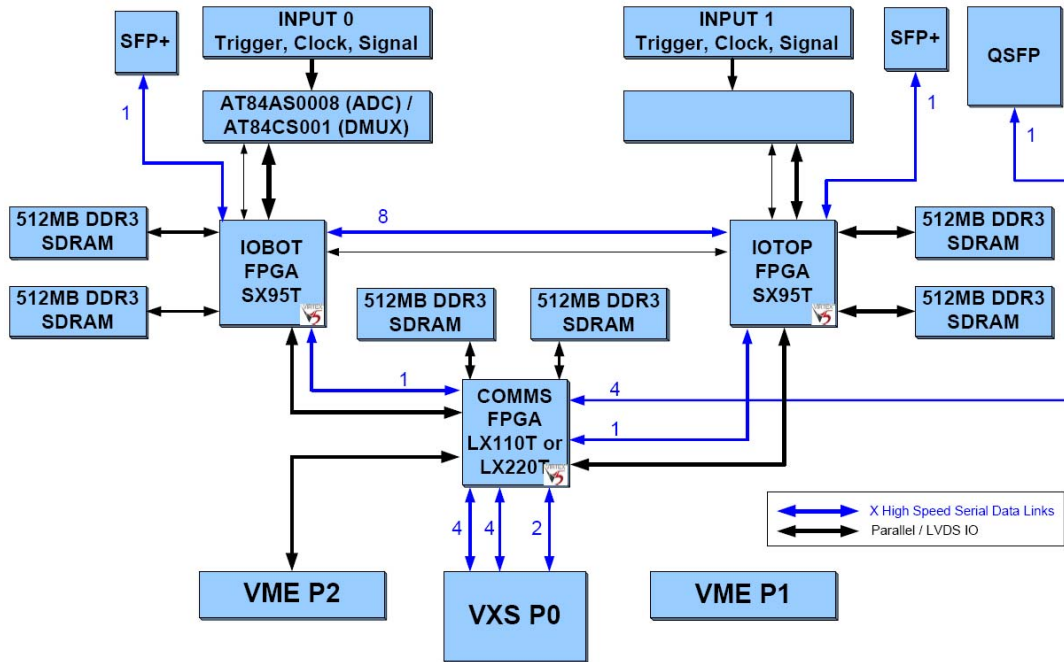


Figure 5.1: Block diagram of Neptune-V5 VXS board, figure from [4]

Together with the hardware a software development kit is also available. One of the components in this kit is an example software/firmware to capture data samples from the ADC and to download these to the computer via the Ethernet link using a proprietary Quixilica Quixstream API. This software/firmware is called 'Data Capture'. A block diagram of the data capture FPGA firmware is shown in *Figure 5.2*. This firmware has the ability to capture data from two ADC:s, one connected to the IOBOT FPGA and the other (if fitted) to the IOTOP FPGA. In such a configuration essentially the same firmware is loaded into both FPGA:s and a master FPGA is negotiated automatically. The FPGA with an Ethernet module in the SFP+ slot is elected master. To be able to process data in the FPGA at high sample frequencies further demuxing is needed. The Data capture firmware uses a second 1:4 DEMUX built into the ADC/DEMUX interface resulting in an internal data rate at a sixteenth of the sample frequency. This means that at the maximum sample rate of 2.2 GS/s the FPGA logic needs to operate at 137.5 MHz. The samples are sign-extended to 16 bits so the PC-software can operate with short integers. In that operation the tag bit is simultaneously encoded into one of the most significant bits. The tag bit indicates the first sample after the external trigger event if used. It is also possible to use software triggers via the PC software. Also clear from *Figure 5.2* is that DDR bank 1 is used to store samples from the ADC connected directly to the FPGA while DDR bank 0 is used to store the data from the second ADC (optional) connected to the other IO FPGA. Before the data is sent to the PC through the Quixstream core the byte order is reversed to ease the burden on the PC which use little-endian format (x86 architecture). [5]

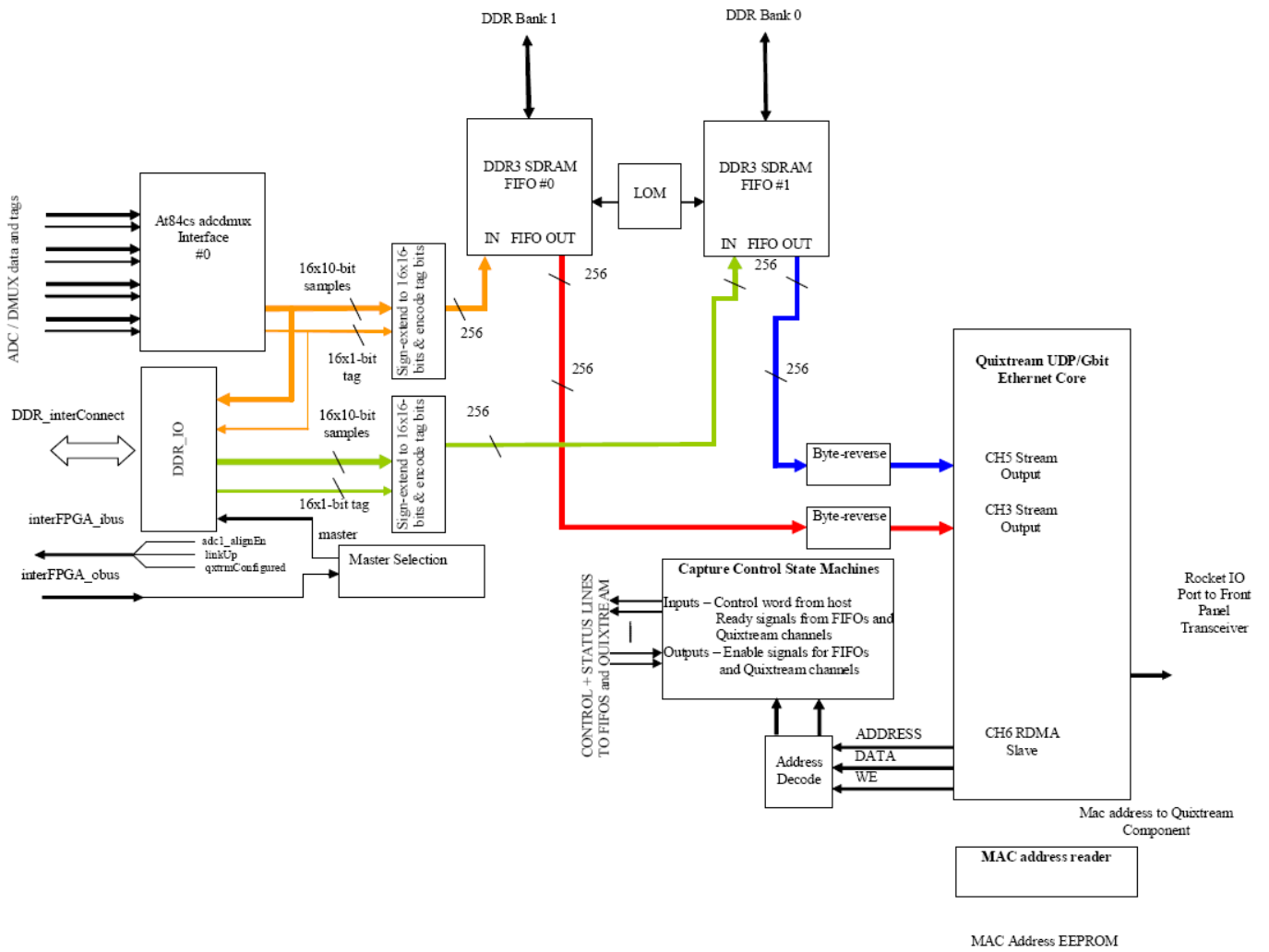


Figure 5.2: Block diagram of data capture FPGA firmware, figure from [5]

5.2 Implementation

This work will use the architecture from the Data Capture firmware as a base to implement the proposed Galileo down converter. As always with hardware design there is a tradeoff between hardware cost and speed. The faster sampling rate needed the more the hardware needs to be parallelized. To run the FPGA at f_s and operate on one sample/cycle is very difficult to achieve. This design was chosen to operate with four samples per cycle at the input. This means that the design has to work at a frequency of $f_s/4$ and this is a good tradeoff point. To avoid rewriting the ADC/DEMUX interface and spend time with probable timing issues at the inputs it was chosen to place a 4:1 MUX and a 1:4 DEMUX around the down converter. This will make it possible to use an internal frequency and operate the down converter at $f_s/4$ while the rest of the design is running at $f_s/16$. Since the hardware used in the project does not have a second ADC and the project would have no use for that the DDR bank 0 is used to store the output of the down converter before it transfers to the PC. A block diagram of the implementation is shown in *Figure 5.3*.

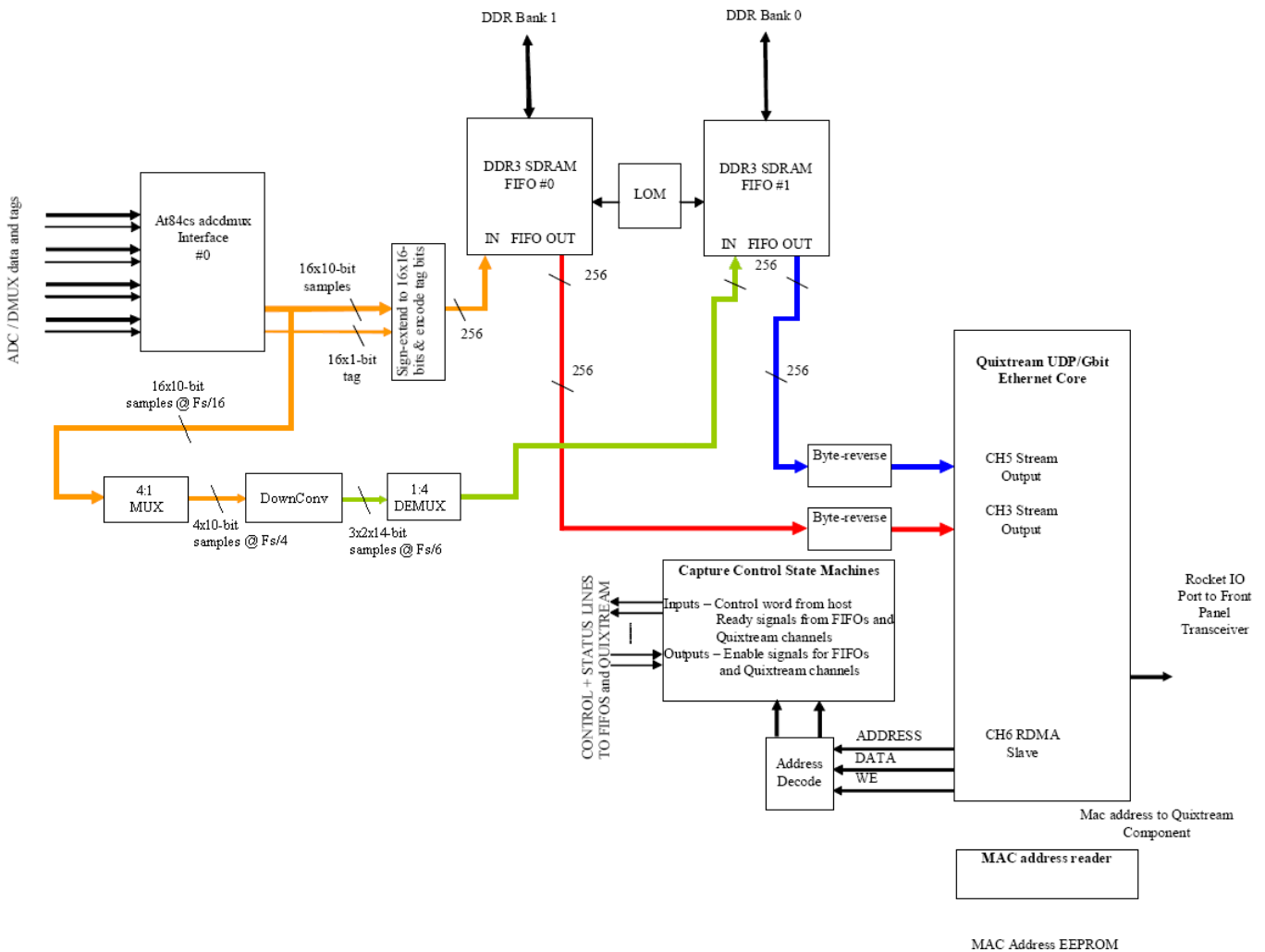


Figure 5.3: Modified data capture firmware block diagram

The implementation is written with ASIC implementation in mind so in the DownConv block no Xilinx specific code is used. This means that no DSP slices are used which will increase the ratio of the general logic used.

5.2.1 Mixers

The mixers are built with a sine/cosine look up table, bit shifting and adders. A block diagram of a mixer can be seen in *Figure 5.4*. The mathematical operation that needs to be implemented is

$$(I + jQ) \cdot (\cos + j \sin) = (I \cdot \cos - Q \cdot \sin) + j(Q \cdot \cos + I \cdot \sin) \quad 5.2.1$$

where $(I + jQ)$ is the incoming sample and $(\cos + j \sin)$ is the complex mixer frequency. The multipliers are multiplying the incoming samples which are in two-complement form with the sine/cosine values. The sine/cosine values are stored as positive numbers but with a bit indicating if the last operation is addition or subtraction. The multipliers are built with bit shifting, and-gates and adders arranged as a binary tree. In this case the number of adders for each multiplier is 8 and since they are arranged as a binary tree the depth is $\log_2(8) = 3$.

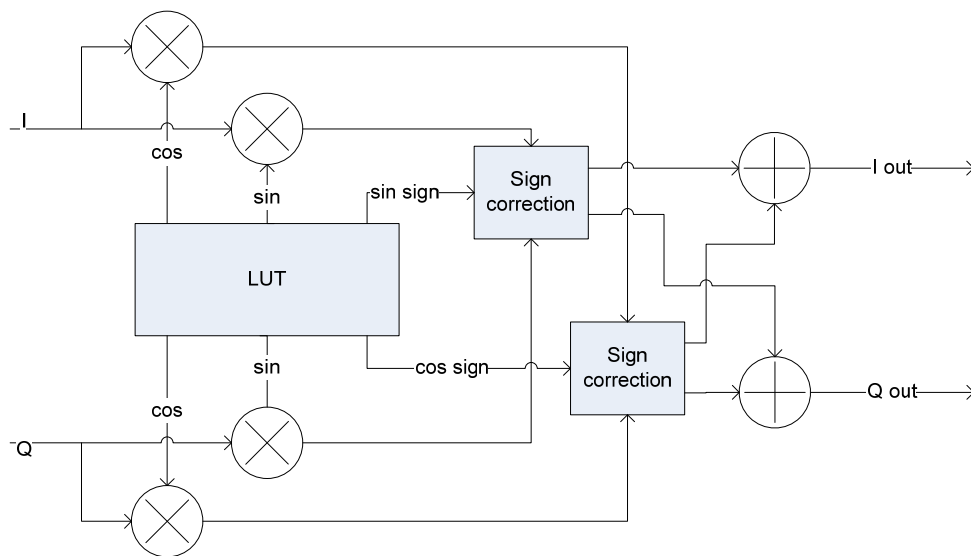


Figure 5.4: Mixer block diagram

5.2.2 Filters

The filters are implemented as decimating FIR filters where the decimation is done at the input with a shifting buffer. Depending on the input rate the shifting buffer must be constructed differently. In *Figure 5.5* a block diagram of a filter with a downsampling factor of 3 and an input rate of 2 samples/cycle is shown as an example (stage 2 filter). Since the input rate differs from the downsampling factor a state machine (with 3 states) is handling the input to the buffer. Then the buffer itself is shifting the samples three steps at an average of every third cycle meaning the filter calculates only every third filtered sample. The fact that the downsampling is done before the filter calculations saves 50% of the logic minus the state machine. If the filter was to downsample after filtering two identical filters (the other with an offset of one sample) would be needed due to the input rate of 2 samples/cycle. The multiplications with the constant filter coefficients are done with shiftings and additions/subtractions. If subtraction is needed the bits are inverted and fed into the adder tree. This will not give the 2-complement that is wanted so a constant is added to the tree to complement all inverted numbers at once. In the example in *Figure 5.5* four numbers are inverted meaning that the constant at the end should be 4. The binary tree of adders is pipelined where needed. The output of the filter is scaled with a factor to utilize the bits more efficient. If the re-quantization after the filter is outputting 14 bits it is a waste if the filtered output would only use typically 13.4 bits. The filter scaling factor has been decided with the help of MATLAB simulations to use the bits most efficient while still keeping a margin of 1 bit (~6 dB) after the re-quantization to avoid saturation. The filter scaling factors can be found in *Table 3.6*.

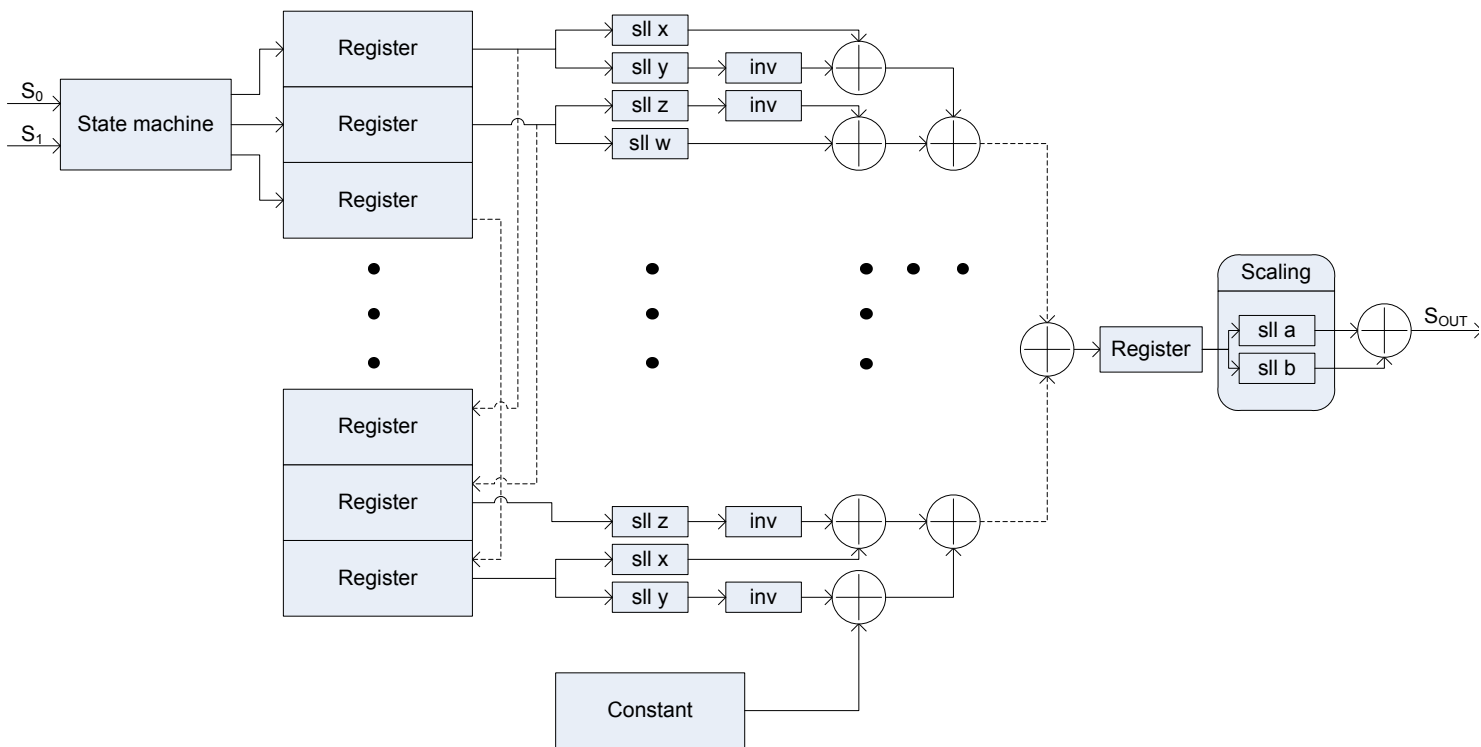


Figure 5.5: Filter block diagram

5.2.3 Re-quantization

The re-quantization rounds and discards a predefined number of bits. The rounding is done by adding 0.5 to the LSB and then truncating the unwanted least significant bits. To not waste hardware some of the most significant bits are also discarded. To see exactly which bits are kept please see *Table 3.5*. In *Figure 5.6* a block diagram of a re-quantizer with output length of $Y+1$ bits that truncates X most significant bits is shown. (This means that the bit representing 0.5 will be the bit at length of the incoming sample minus Y).

It is important to round symmetrically to not introduce any bias. To achieve that the incoming negative numbers will be negated and then treated the same as the positive numbers until the output where they will be negated again. Here the adder to add the 2-complement is also used to add the "0.5". If the number was positive then it just adds the corresponding 0.5 and if the number is negative it adds the corresponding 0.5 and 1. To not introduce extra errors it is also necessary that the numbers are not allowed to overflow but are being saturated instead. The hardware checks the most significant bits that are to be truncated to see that they are all zeros. If they are not (the result of the comparison is zero) it means the number has grown into the bits that are to be thrown away so the biggest possible value are outputted instead.

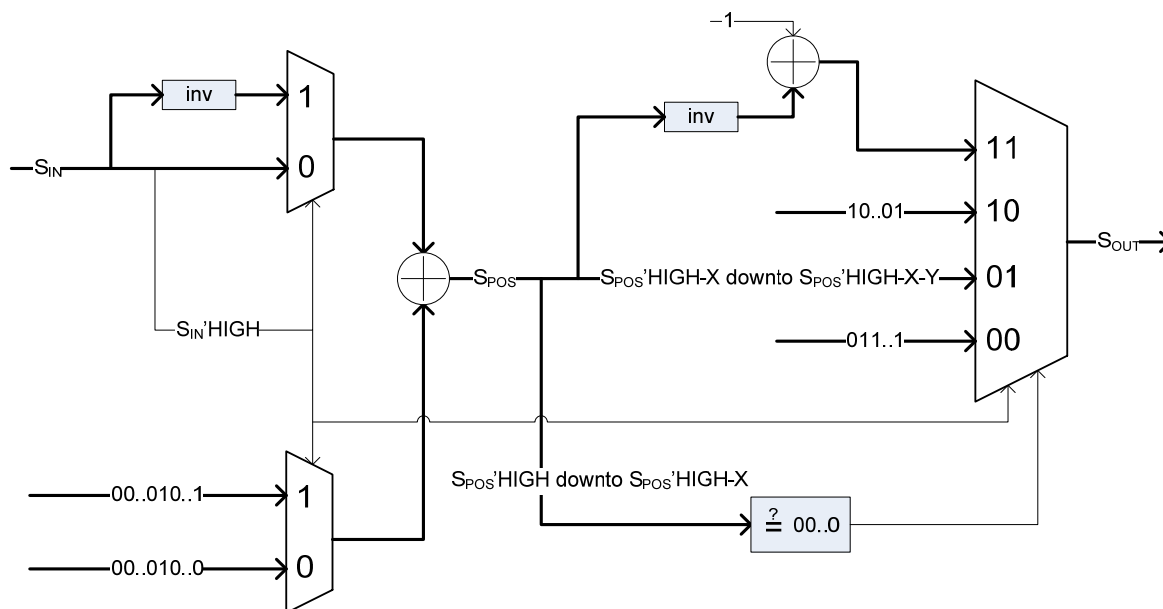


Figure 5.6: Re-quantizer block diagram

6 VERIFICATION

Verifying hardware is a non trivial task and can be very time consuming. In this project it was chosen to implement a bit true MATLAB model to compare with the hardware implementation. The bit true MATLAB model is accompanied by a floating point model which the function of the bit true model can be compared against. That comparison has been made visually.

6.1 Methodology

As shown in *Figure 1.2* both the unprocessed RAW samples from the ADC and the output of the VHDL implementation can be downloaded to the PC. This means that the data from the ADC can be processed with the bit true MATLAB model and the result can be compared with the output of the VHDL implementation in the FPGA.

6.2 Result

The test cases run and the result is shown in the *Table 6.1*.

Table 6.1: Test cases run

| Input signal | Test covers | Result |
|------------------------------|--------------------|---------------|
| 1540f ₀ 136 mV pp | E1 signal path | Passed |
| 1165f ₀ 124 mV pp | E5 signal path | Passed |
| 1250f ₀ 132 mV pp | E6 signal path | Passed |

7 PERFORMANCE ANALYSIS

7.1 Noise error in ADC

The ADC has several sources of unavoidable errors [6]. One intrinsic error is quantization error which is generated due to the finite resolution of the ADC and is a fundamental limitation of an ADC. The amplitude of the quantization error at the sampling instant is between zero and half of one quantization step Δ and has a uniform distribution. For a 10-bit ADC, Δ is,

$$\Delta = \frac{X_{FS}}{M} = \frac{X_{FS}}{1024} \quad (7.1.1)$$

Where X_{FS} is the input range of the ADC and M is the number of quantization levels, which in this case is $M=1024$.

Since all values in the interval $n\Delta$ to $(n+1)\Delta$ are quantized to the mid-point $(n+1/2)\Delta$, the quantization error is in the range $-\Delta/2 \leq \epsilon_Q \leq \Delta/2$. If one of the two edges, $n\Delta$ or $(n+1)\Delta$, is used to represent the quantization interval, then the range of ϵ_Q would shift up or down by $\Delta/2$. The range of ϵ_Q becomes $0 \leq \epsilon_Q \leq \Delta$ and $-\Delta \leq \epsilon_Q \leq 0$ for utilizing lower or upper edges, respectively. Nevertheless, all these cases indicate that the dynamic range of the quantization error ϵ_Q is Δ . Moreover, by assuming that the probability distribution function $p(\epsilon_Q)$ of ϵ_Q is a constant and would become 0 outside of the range $-\Delta/2 \leq \epsilon_Q \leq \Delta/2$, the probability distribution function of ϵ_Q is

$$\begin{aligned} p(\epsilon_Q) &= \frac{1}{\Delta} \quad \text{for } \epsilon_Q \in -\Delta/2 \dots \Delta/2 \\ p(\epsilon_Q) &= 0 \quad \text{otherwise} \end{aligned} \quad (7.1.2)$$

Consequently, the power of ϵ_Q can be estimated as

$$P_Q = \int_{-\infty}^{\infty} \epsilon_Q^2 \cdot p(\epsilon_Q) d\epsilon_Q = \int_{-\Delta/2}^{\Delta/2} \frac{\epsilon_Q^2}{\Delta} d\epsilon_Q = \frac{\Delta^2}{12} \quad (7.1.3)$$

For an ideal sampling, samples are taken at the exact sampling time. But in a real situation, the sampling-time is affected by the uncertainty of the clock and also the unpredictable delay between the logic. The sampling jitter affects the value of the sampled signal by an error that depends on both the jitter and the time derivative of the input. For a sine wave

$$X_{in}(t) = A \cdot \sin(\omega_{in} t) \quad (7.1.4)$$

the error $\Delta X(nT)$ is given by

$$\Delta X(nT) = A \cdot \omega_{in} \cdot \delta(nT) \cos(\omega_{in} nT) \quad (7.1.5)$$

where $\delta(nT)$ is the sampling of a random variable $\delta_{ji}(t)$ and the error $\Delta X(nT)$ is the sampling of

$$x_{ji}(t) = \delta_{ji}(t) \omega_{in} X_{in}(t) \quad (7.1.6)$$

Assuming the spectrum of $\delta_{ji}(t)$ is white then the spectrum of $x_{ji}(t)$ is also white as cosine modulation has no effect on white spectra. Therefore, the jitter noise is also considered to be white. The power of the jitter error $x_{ji}(t)$ is given as

$$P_{j_i}(t) = [\delta_{j_i}(t)\omega_{in}X_{in}(t)]^2 = \frac{A^2\omega_{in}^2}{2}\delta_{j_i}(t)^2 \quad (7.1.7)$$

Last but not the least an often considered non-ideal characteristic of the ADC is kT/C noise. It occurs in all real sampled data systems due to the unavoidable thermal noise and the physical imperfections of the circuits. Obviously, the noise power contributed by kT/C noise is

$$P_{kT/C} = \frac{kT}{C_S} \quad (7.1.8)$$

where k = Boltzmann constant 1.38×10^{-23} J/K
T = the absolute temperature in Kelvin
C_S = the sample and hold capacitance

From *Equation (7.1.8)*, the noise only goes to zero for zero temperature and infinite sample and hold capacitance. That is the reason why kT/C noise is also referred as a fundamental limit of data convertor.

All these non-linearities mentioned above not only reduce the dynamic range of the signals that can be quantized by the convertor but also reduces the effective resolution of the ADC. However, the non-linearity errors can sometimes be compensated for by calibration, or prevented by testing at later stages.

7.2 ADC Non-linearity modeling

7.2.1 Introduction

As one of the key components in a signal processing system, the ADC plays a fundamental role in interfacing the processing core to the analog world. A good performance ADC requires high linearity and small quantization errors. The following test is carried out to investigate the high-order nonlinear errors (section 7.2) and non-linear phase distortion (section 7.3) of the ADC. The non-linear behavior, such as harmonic and inter-modulation distortion, of the ADC can introduce a whole host of spurious signals into the spectrum. Moreover, it is important to know the phase distortion of the ADC since the received signal is phase modulated.

7.2.2 Test method

One common way of modeling an ADC's non-linearity behavior is to use a sine fitting method together with a lookup table model strategy. First, the tested ADC measures a sine wave generated from the synthesizer. Then a second sine wave is modeled in the software to fit into the measurement. Finally the lookup table (LUT) can be generated by storing the differences in values between measurement and model. Instead of memorizing the input-output relation, the input-noise relation is being estimated and memorized in the LUT. The noise level suggested by the LUT will be used to correct the input signal to find the corresponding output signal. The LUT model, used in this test, does not take into account the physical realization of the ADC at all. The device is characterized by input-noise analytical or numeral relations, without going deep into the internal structure. Moreover, the LUT is easy and fast to implement using software such as MATLAB. However, for different ADC's, each aleatory variation of model parameters requires the generation of a new table. Thus, the LUT model has minimum flexibility [7].

The aforementioned strategy turns out to be effective in the verification and is carried out in the following test.

7.2.3 Lookup Table modeling and analysis

As an example, a signal y with frequency $f_c=1535f_0$ has been generated from a signal synthesizer and measured via the NeptunV5 card. A MATLAB code has been developed to analyze the measurement and generate a lossless signal $y'=A\sin(2\pi f_c t+\varphi)+\varepsilon$, where the amplitude A , phase φ and offset ε are extracted from the measurement, to fit into the measurement. Consequently, y' is considered as an ideal output from the synthesizer and used as a model in the following analysis. Both measurement and model signal are shown in *Figure 7.1*.

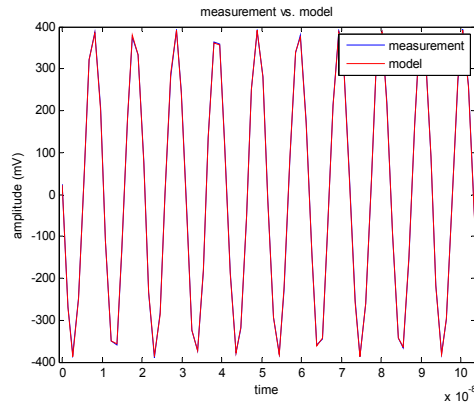


Figure 7.1: Comparison of model sine wave and measurement in the time domain

Although both the transmission wires and the hardware on board add noise to the signal, the noise power is dominated by the non-ideal characteristics of the ADC. This noise can be estimated from the difference between measurement and model as $dy=y-y'$. The estimated noise is shown in *Figure 7.2*. It can be seen that the total error caused by non-ideal characteristic of ADC is approximately 4% of the full signal range ($\pm 512\text{mV}$).

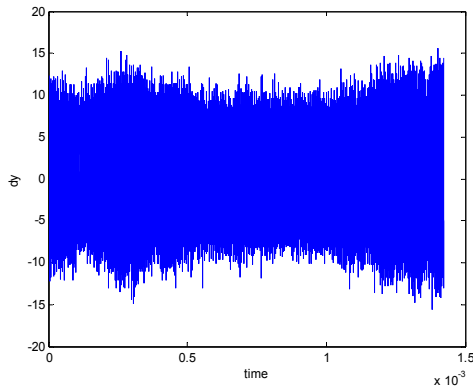


Figure 7.2: ADC noise

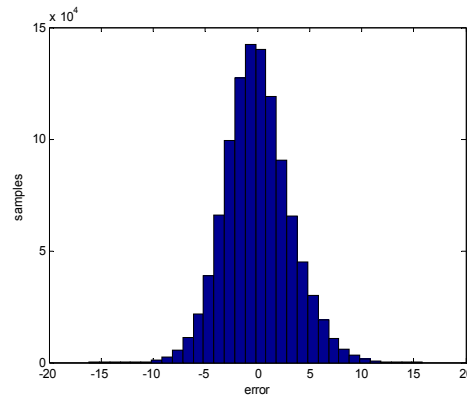


Figure 7.3: Histogram of noise distribution (unit is LSB for x-axis)

The distribution of noise is Gaussian, as shown in the histogram of *Figure 7.3*, the mean value equals zero and the standard deviation is approximately $\sigma_e = 3.1$ LSB. As discussed above, the quantization noise has a uniform distribution, but both clock jitter and kT/C noise are Gaussian distributed. Therefore quantization noise is not dominating the result. In this case, it is only the clock jitter and the kT/C noise that matters. Assuming that the major noise power comes from clock jitter, from *Equation (7.1.7)* the clock jitter is calculated as,

$$\delta_{ji}(t) = \frac{\sqrt{2P_{ji}(t)}}{A \cdot 2\pi f_c} = \frac{\sqrt{2}\sigma_e}{A \cdot 2\pi f_c} \quad (7.2.1)$$

where $\sigma_e = 3.1$ LSB, is the standard deviation of jitter noise
 $A = 512$ LSB, is the full signal range
 $f_c = 1535f_0$, is the carrier frequency of examined signal.

Equation (7.2.1) yields the jitter time $\delta_{ji}(t) \approx 0.87$ ps-rms which is small enough compared to the sampling rate and can be accepted in this project.

As shown in Figure 7.4, the plot of the corresponding output along with the input signal displays a straight red dotted line. However, affected by the non-linear characteristics of the ADC, such as clock jitter, the measurement data is inter-dispersed around the ideal sampling value as reflected by blue dots in Figure 7.4. Moreover, the same figure shows the limitation of the chosen $f_c = 1535f_0$. In this analysis, although f_c has been carefully chosen; to cover as much of the quantization levels as possible, it still has the inadequacy for generating the Look up table (LUT) as will be stated in the following part.

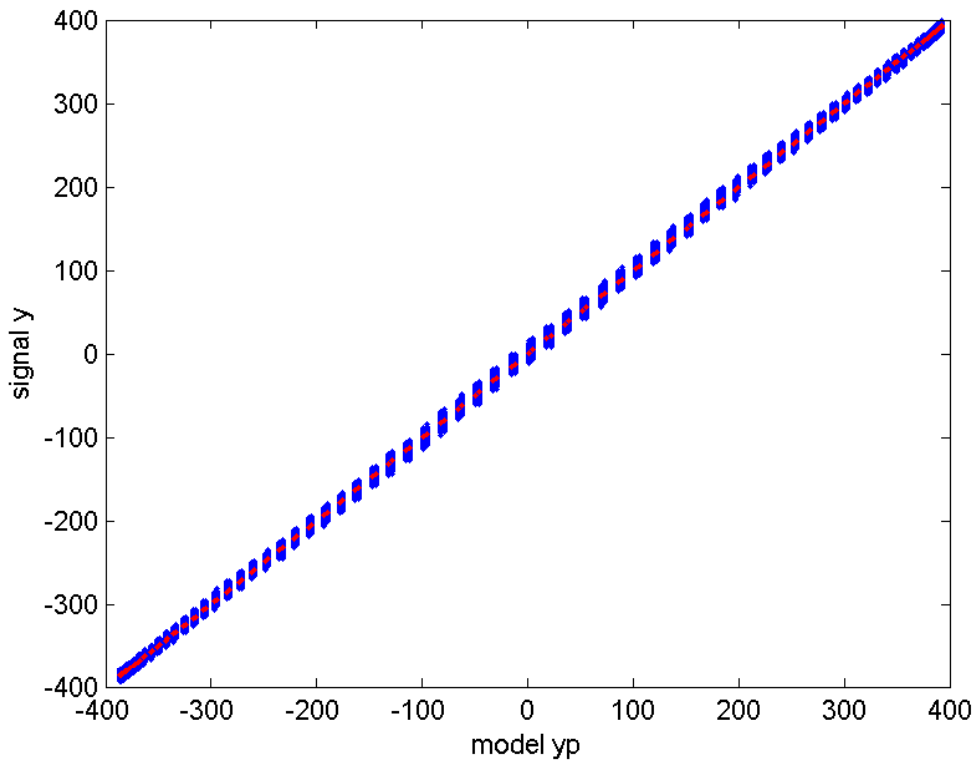


Figure 7.4: Ideal and real case input-output relationships (unit is LSB for both axis)

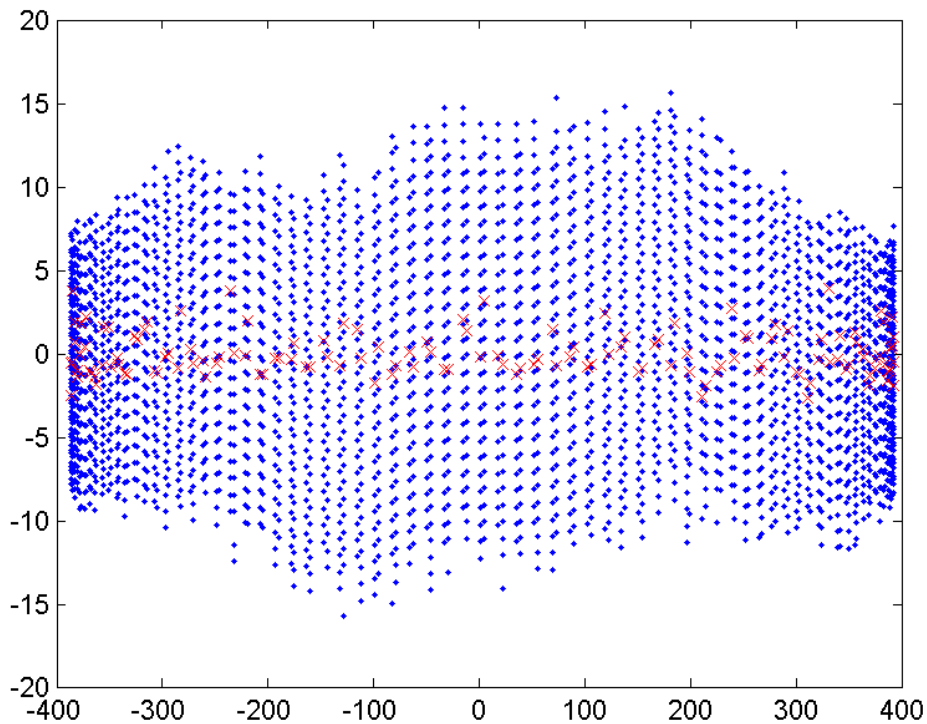


Figure 7.5: Quantization level vs. noise relationship (unit is LSB for both axis)

In Figure 7.5, the noise dy is plotted along with the quantization levels as the blue dots shown. The mean value of non-linear noise at each particular quantization level is calculated as shown by the red crosses. The curve suggested by the red crosses is the estimated nonlinear transfer function for this specific frequency $f_c=1535f_0$. For an ideal case, the nonlinear transfer function should be frequency independent so that one LUT can be used for all frequencies of the E1, E5 and E6 bands. However, because of some frequency related noise factors, such as clock jitter, the transfer function in this test is frequency dependent and changes for different input frequencies. To find a transfer function that can fit well for all frequencies in the three bands, a method that combines several different frequency transfer functions has been used. The transfer functions for the center frequencies of each band, E1, E5 and E6, are generated. Another three frequencies of 1578.86 MHz, 1190.44 MHz and 1276.75 MHz, which falls in the bands E1, E5 and E6 respectively are carefully chosen and the transfer functions related to these three frequencies are generated. The reason to choose these frequencies is because they are able to cover most of the quantization levels after the ADC. Finally, the six particularly selected transfer functions are combined and stored in a LUT for later analysis.

The specific transfer function for the frequency $1535f_0$ and the general case are illustrated by the blue curve in Figure 7.6a and Figure 7.6b respectively.

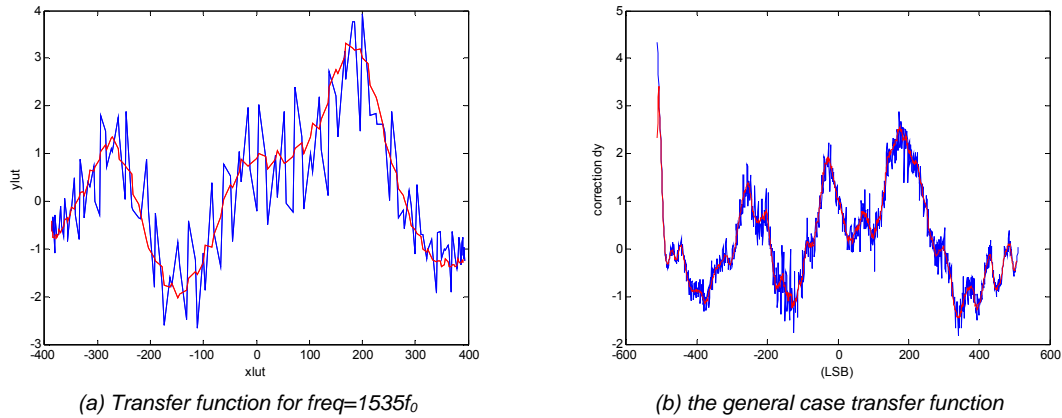


Figure 7.6: Illustration of transfer function (unit is LSB for both axis)

However, in both case, the combined transfer function is very noisy as shown by the blue curve in *Figure 7.6*. To smooth the curve, a 10-tap averaging filter is used to filter away the noise. After filtering, the first 5 samples of the output transfer function are trimmed off due to the delay introduced by the filter. The red curve in *Figure 7.6* shows the transfer function after filtering and trimming. As can be seen from the plot, the new low-noise transfer function is aligned with the original transfer function but has 5 samples less. The steep slope at the left end of the curve, in *Figure 7.6b*, together with the missing 5 samples seems risky and not easily to believe; whereas this pattern will not affect the result due to the method used to generate this LUT, which constrained the valid range of the table from -490.96 LSB to 497.92 LSB.

7.2.4 ADC non-linearity modeling with LUT

The ideal sine wave model was interpolated by passing it through the LUT generated previously and then comparing it to the measurement in the frequency domain. The setup of the system is shown in *Figure 7.7*.

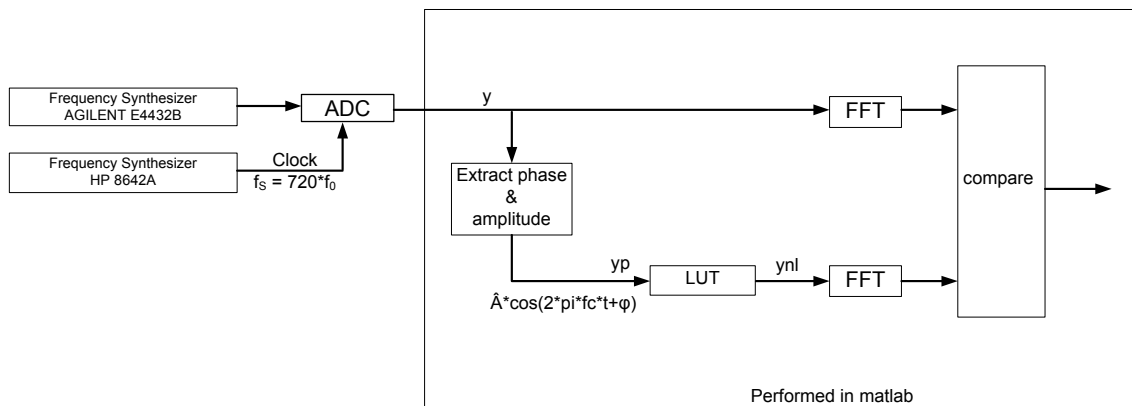
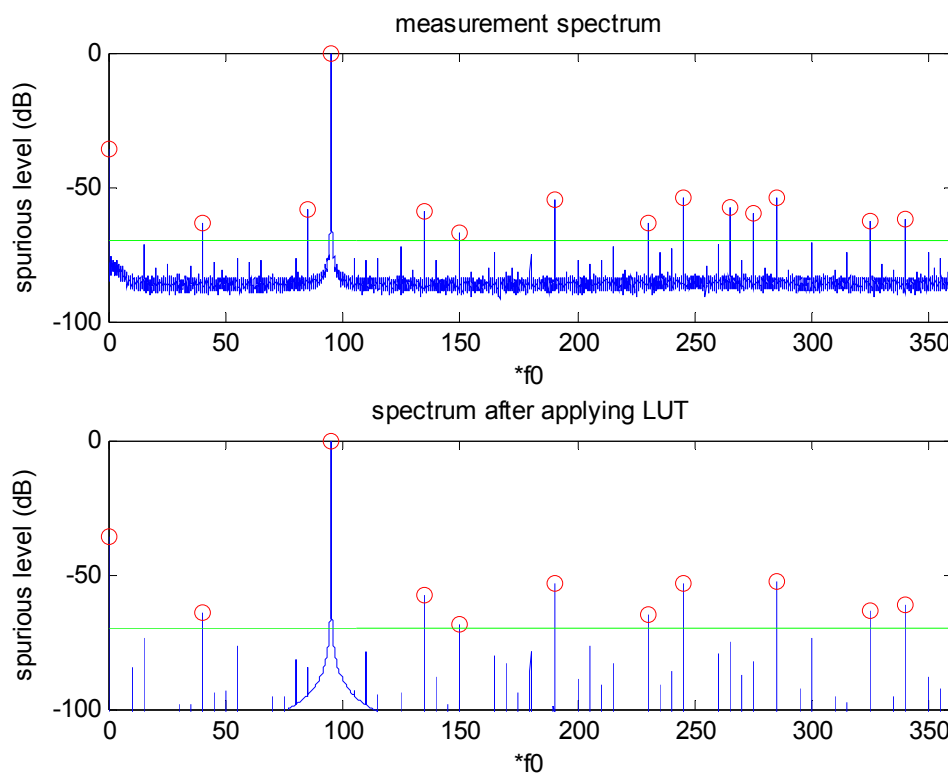


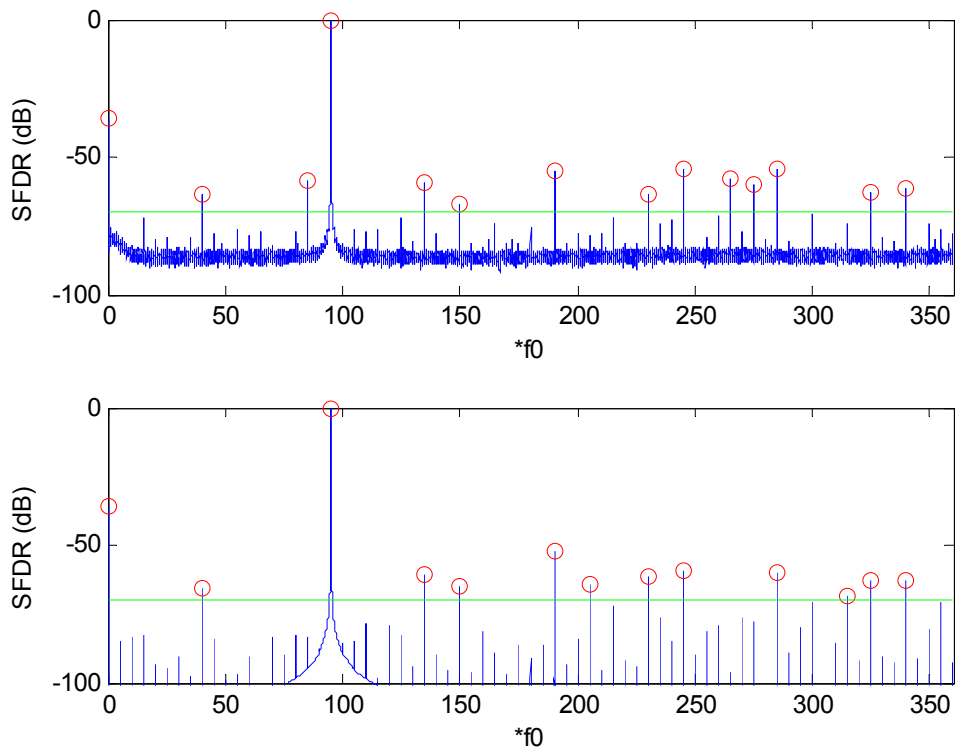
Figure 7.7: Testing setup for LUT model

Figure 7.8a and Figure 7.8b show the comparison results between the model and the measurement by applying a specific LUT at $1535f_0$ and a general case LUT. The green line demonstrates the SFDR level. Only spurs above -70 dB are interesting and has been marked with red circle in the plot. The upper and lower subplot yields the spur distribution of the measurement signal and the model signal after LUT, respectively. Comparing the two subplots, most of the spurs can be explained by the harmonics after applying the LUT but some of them are still missing in the modeled spectrum. Moreover, in the subplots reflecting the results from the general case model, Figure 7.8b, some of the spurs appear at unexpected frequencies in the modeled spectrum but these spurs are below the SFDR level in the measurement signal spectrum. The cause of false spurs in the general case model is the frequency-depend characteristic of the LUT. Although a lot of effort has been devoted to select the signal frequencies that are suitable for generating the LUT, it can not completely eliminate the frequency dependant properties of the LUT but only to diminish it to the maximum extent. This is a trade off between the complexity of generating the LUT and the accuracy one can obtain.



(a). Comparison of results with transfer function for $1535f_0$

Figure 7.8: Results after applying LUT



(b). Comparison of model and measurement with general LUT

Figure 7.8: Results after applying LUT

7.2.5 ADC non-linearity modeling including spurs from ADC

To further examine the lost spurs in the model spectrum, inter-modulation of the measurement has been studied. Inter-modulation is the result of two or more signals of different frequencies being mixed together, forming additional signals at frequencies that are the sum and difference of those frequencies. This phenomenon is also generated by the non-linearity of the ADC circuit which contains analog components. One possible reason to explain the inter-modulation in this case is to consider the affect of the sampling clock $f_s=720f_0$. Since the input of the NeptuneV5 card has a 1:4 DEMUX, the sampling clock frequency may also have been affected and generate a frequency of $f_s/4=180f_0$. This signal leaks back and mixes with the input signal and its harmonics because of the unexpected characteristic of the analog circuit. Besides the affects of the frequency $f_s/4$, another hypothesis frequency is $f_s/2$. The setup block diagram of the test is illustrated in *Figure 7.9*. The model signal is added with the two hypotheses and then passed through the LUT.

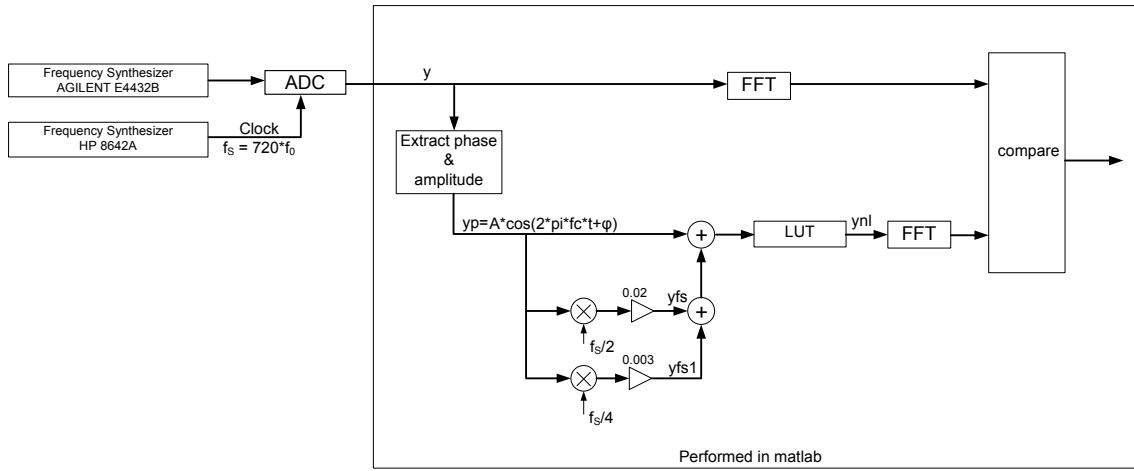
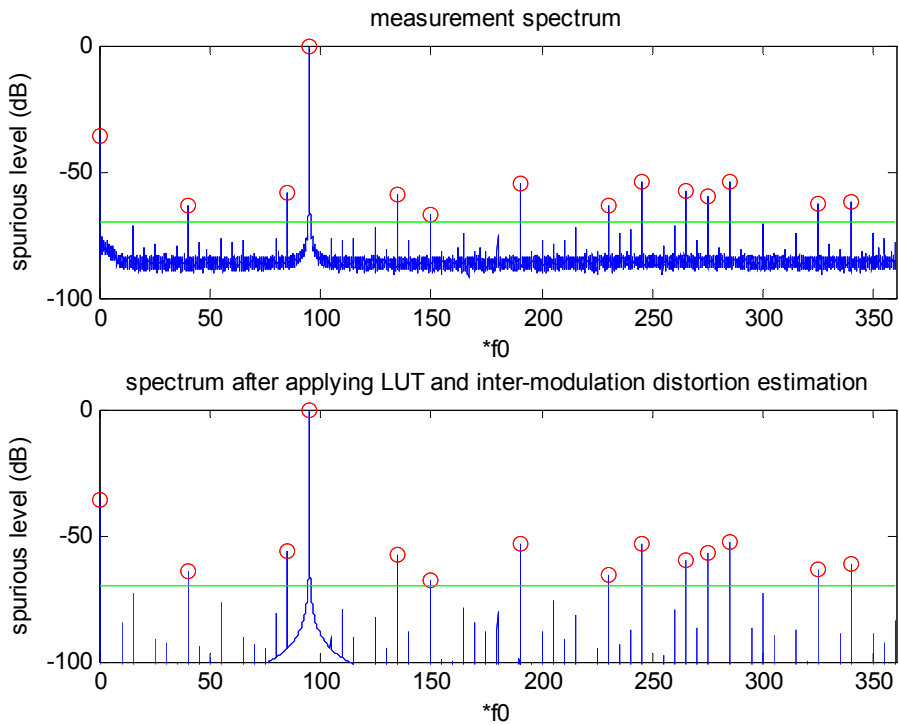


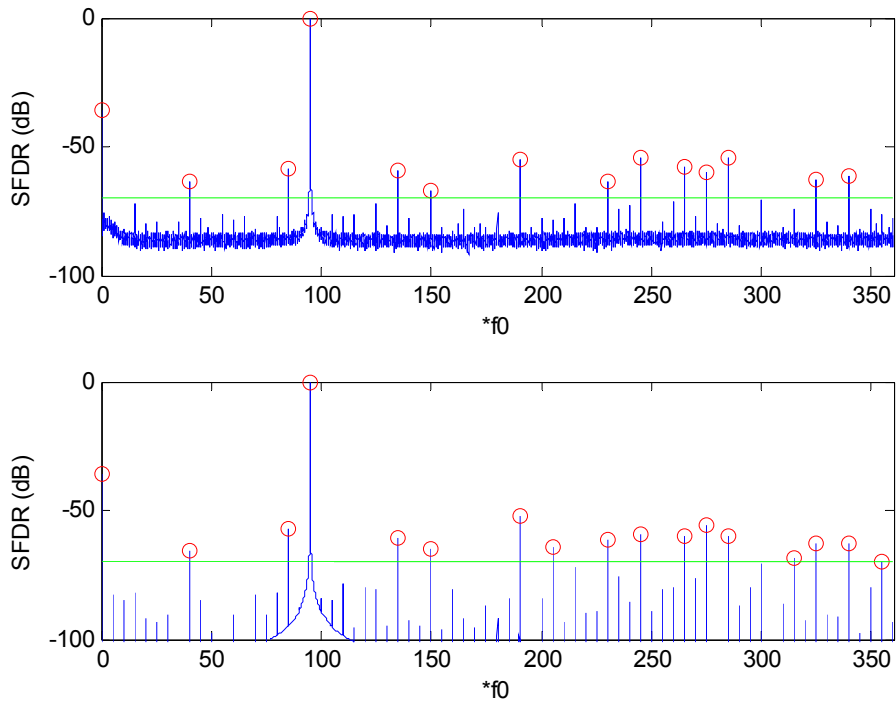
Figure 7.9: Testing setup for inter-modulation model

The result is illustrated by *Figure 7.10a* and *Figure 7.10b*.



(a) with LUT for $1535f_0$

Figure 7.10: Comparison of model and measurement after applying inter-modulation distortion estimation.



(b) with general LUT

Figure 7.10: Comparison of model and measurement after applying inter-modulation distortion estimation.

When applying the specific LUT generated for the measurement frequency, *Figure 7.10a* confirmed the assumption made above that all the spurs generated from the ADC in the measurement can be explained after applying inter-modulation distortion to the model. On the other hand, *Figure 7.10b* also confirmed the results but the inherent limitation of the frequency-dependant property of the LUT restrained the performance.

7.2.6 Results and conclusion

Table 7.1 lists the test results reflected in Figure 7.8 and Figure 7.10. From Table 7.1, the conclusion can be made that the model using the specific LUT at $1535f_0$ and with inter-modulation distortion has the best performance.

Table 7.1: Summary of modeling results

| Measurement method Frequency (* f_0) | Measurement [dBc] | Applying LUT | | With LUT and inter-modulation | |
|---|----------------------|-------------------------------|-------------------------|-------------------------------|-------------------------|
| | | LUT for $1535f_0$ [dBc] | General LUT [dBc] | LUT for $1535f_0$ [dBc] | General LUT [dBc] |
| 0 (DC) | -35,8 | -35,8 | -35,8 | -35,8 | -35,8 |
| 40 | -63,6 | -64,3 | -65,4 | -64,0 | -65,4 |
| 85 | -58,6 | -84,1 | -83,3 | -56,2 | -56,7 |
| 135 | -58,9 | -57,8 | -60,5 | -57,7 | -60,6 |
| 150 | -67,3 | -68,3 | -65,0 | -68,0 | -64,9 |
| 190 | -54,8 | -53,0 | -52,0 | -53,0 | -52,0 |
| 205 | -78,6 | -76,7 | -64,1 | -76,0 | -64,1 |
| 230 | -63,6 | -65,2 | -61,2 | -65,6 | -61,3 |
| 245 | -54,3 | -53,2 | -59,0 | -53,3 | -59,1 |
| 265 | -57,5 | -75,0 | -96,0 | -60,1 | -60,2 |
| 275 | -59,9 | -82,4 | -78,0 | -57,0 | -55,9 |
| 285 | -54,0 | -52,4 | -60,0 | -52,3 | -60,0 |
| 315 | -74,3 | -97,5 | -68,4 | -87,0 | -68,6 |
| 325 | -62,6 | -63,3 | -62,8 | -63,5 | -62,7 |
| 340 | -61,7 | -61,2 | -62,8 | -61,3 | -62,8 |
| 355 | -76,4 | -92,6 | -70,4 | -92,2 | -69,9 |

In this test, only the second order distortion is taken into consideration. Moreover, although all spurs appearing in the measurement can be modulated with the effort of applying both the LUT and the inter-modulation distortion, it is generally intractable to adjust the inter-modulation level and generate a perfect model. The foremost reason is that it all depends on the analog components and differences in each application. Though the level of inter modulation used in this test was empirically determined and the modeled spectrum is not perfect, the results are still satisfying.

Another noticeable but not crucial demerit of this transfer function method is that it only models the amplitude distortion due to the ADC's non-linear characteristics. Therefore the following test has been carried out to analyze the frequency response of the ADC, which gives the information of the amplitude and phase distortion of the ADC's non-linear characteristics.

7.3 ADC frequency response modeling

7.3.1 Introduction

It is important for the receiver to know the phase distortion since the navigation signal is phase modulated. Wherever amplitude and phase distortion occurs, it should be notified and compensated for at the receiver part. The reasons for the phase distortion of the ADC needs to be looked into; it comes from clock jitter caused by the inaccuracy of sampling clock generator, thermal noise and other circuit properties caused by analog components of the ADC.

7.3.2 Test method and setup

One straightforward way to examine the non-linear phase distortion of the ADC is to use a chirp synthesizer which examines and tests the phase response of the output data directly from the ADC. This method is easy to execute however it sets high requirements on the synthesizer and other equipment used during the test.

Another idea on how to analyze the non-linear phase distortion is to model the ADC frequency response by a known analog filter with the help of a set of measured data from the NeptuneV5 card. The set of measurement data yields the amplitude response of the ADC but gives no information in the frequency domain which is critical to obtain the phase response. The task is to find a filter which has the most similar amplitude response to the ADC so that the non-linear phase distortion can be derived from the filter coefficients. However, a potential risk exists in this method and may confuse the decision because more than one type of filter may be found suitable and fit the amplitude response well. Filters with similar amplitude behavior in the desired frequency band are not ensured to have the same poles and zeros so that the phase response may behave very differently in the rest of band. Whereas the test bands of interest (E1, E5 and E6) are only a very small portion of the analog filter bandwidth it is believed that all analog filters found in this case have similar frequency response in the tested bands and this method remains valid and efficient.

7.3.3 Analysis and result

Due to the affection of non-ideal characteristics of the ADC, the measured curve seems quite nervous and useless especially in the expected bands E1, E5 and E6, as the blue curve shown in *Figure 7.11* indicates. To improve the return loss, a 9dB attenuator has been used at the output of the ADC. The green curve in *Figure 7.11* shows the improved test result, which is more trustable in appearance and will be used throughout the rest of the test. The measured frequency points are distributed in the frequency band 100Hz to 3GHz. More points are tested in the E1, E5 and E6 bands because they are the interesting frequency portions for this thesis work. The desired frequency band is from 0Hz to 1595.88MHz (the upper boundary of E1), therefore a measured frequency up to 3.2GHz is more ideal since that gives twice the margin for the desired frequency band. However due to the limitations of the equipment, only frequencies up to 3GHz are available and has been measured. By experience, the frequency synthesizer in use is not perfect and contains inaccuracies, so a power meter has been utilized to correct for the applied power. Furthermore, a relatively high requirement of accuracy for the measured data is demanded because the whole analysis method is based on this first step. An inaccurate curve can lead in a totally different direction far away from truth. Additionally, one more thing that must be kept in mind is the quality of the wires used to connect the equipment. With an unstable wire, a little twist of the wires between measurements can affect the result and confuse the conclusion. The measured ADC amplitude response up to 3GHz with a 9dB attenuator connected at the output of ADC is showed in *Figure 7.12*.

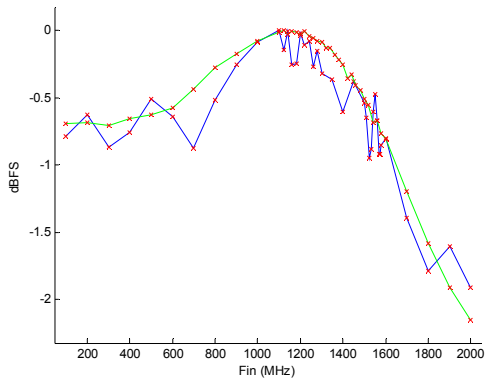


Figure 7.11: Measurement up to 2GHz

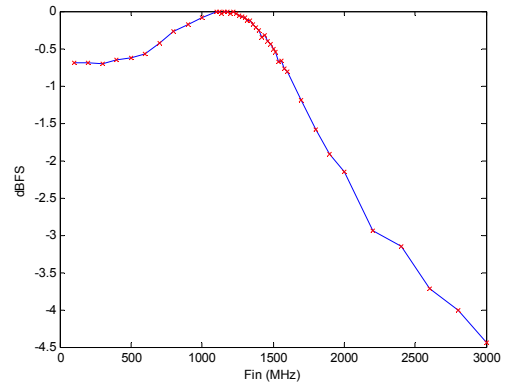
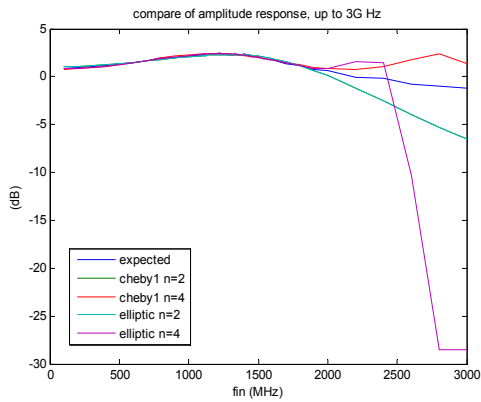
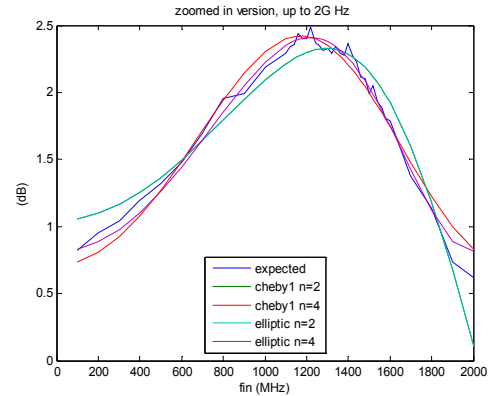


Figure 7.12: Measurement with a 9dB attenuator up to 3GHz

Next, a filter model to represent the measurement curve is investigated. In this case, filters with passband ripple are preferred due to the behavior of the ADC's amplitude response. Consequently, Chebyshev type I and elliptic filters are chosen since they both have similar behavior in the passband. With the help of developed MATLAB code, filters with different types and orders are being investigated. As suggested by Figure 7.13 and Figure 7.14, the cascading of two filters gives better performance than a single filter especially at higher frequencies (2 GHz to 3 GHz).



(a) Right: Model with Cheby I and elliptic filter

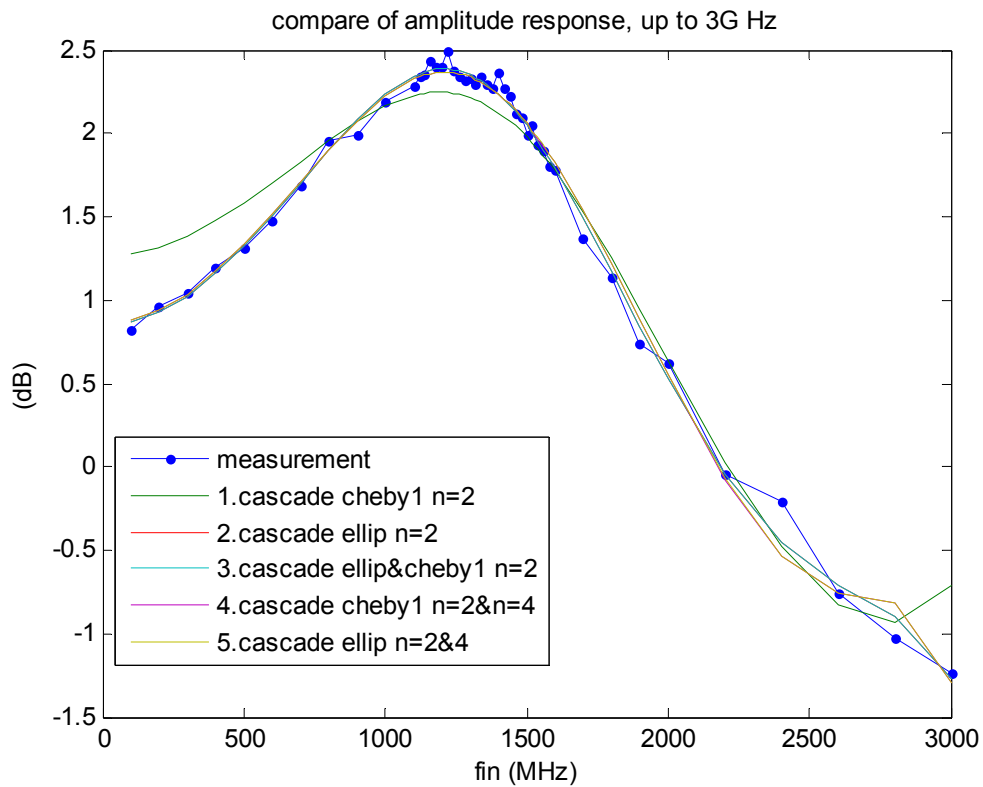


(b) Model with Cheby I and elliptic filter (zoomed in version)

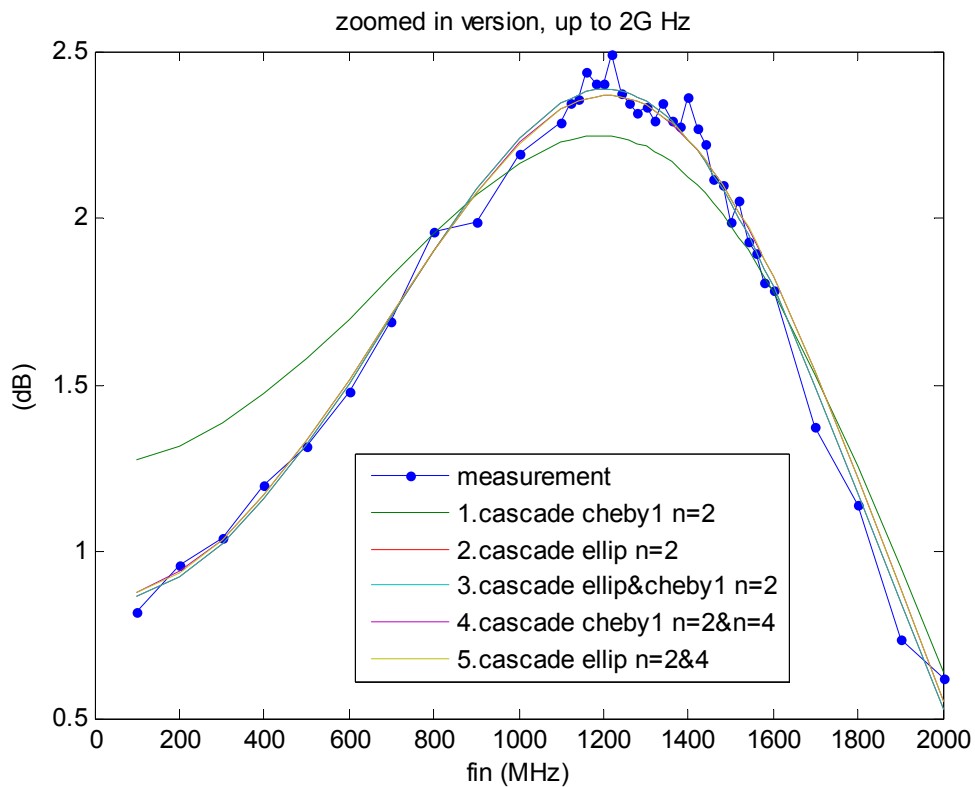
Mapping scheme:

- Blue.....Measurement
- Green..... Chebyshev I, order 2
- Red..... Chebyshev I, order 4
- Cyan..... Elliptic, order 2
- Magenta..... Elliptic, order 4

Figure 7.13: Amplitude response for different filters



(a) Model with cascade filter



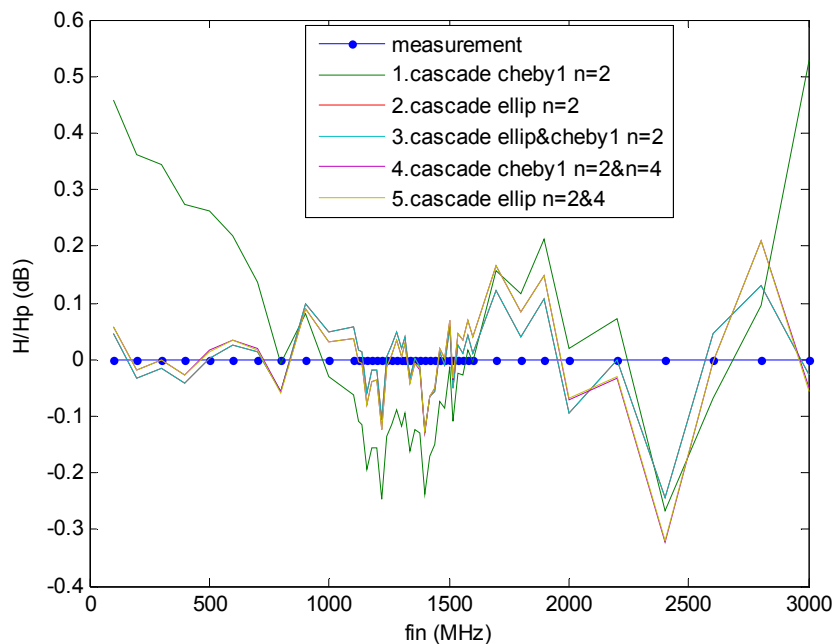
(b) Model with cascade filter (zoomed in version)

Figure 7.14: Amplitude response for different cascaded filters

In *Figure 7.14*,

- The green curve represents alternative 1.
- The red and the cyan curve, that represents alternative 2 and 3 respectively, are on top of each other and therefore can not be seen.
- The magenta and the yellow curve, represents alternative 4 and 5 respectively, are on top of each other and therefore can not be seen.

Except for alternative1, the frequency response curves are very close to the measured amplitude response. Although there may exist other types of cascaded filters, for example a 2nd order elliptic filter and a 4th order Cheby I filter, that fit the curve tightly but the results here are believed to be satisfying for this test. *Figure 7.15* also confirms this conclusion. It shows the comparison result of the ratio between measurement amplitude response and different simulation results in dB, a smaller value indicates that they are more identical to each other.

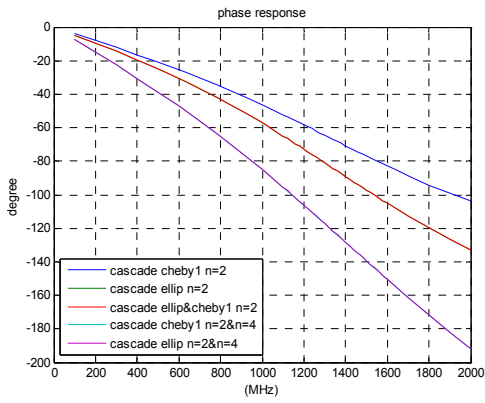


Mapping scheme for Figure 7.14 and Figure 7.15:
 Blue.....Measurement
 Green..... 1. Cascaded Chebyshev I, order 2
 Red..... 2. Cascaded elliptic, order 2
 Cyan..... 3. Cascaded elliptic and Chebyshev I, order 2
 Magenta.....4. Cascaded Chebyshev I, order 2 and 4
 Yellow.....5. Cascaded elliptic, order 2 and 4

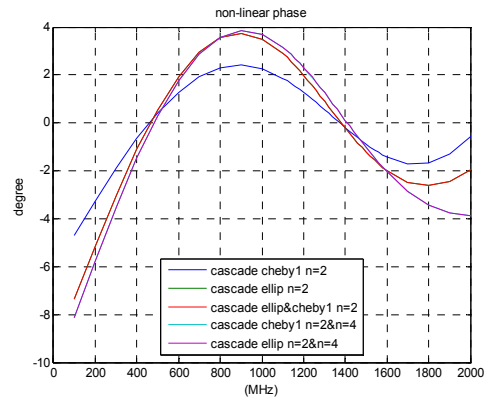
Figure 7.15: Comparison of filter results with measurement

Figure 7.16a and *Figure 7.16b* shows the phase response and the non-linear phase distortion at the measurement band up to 2 GHz.

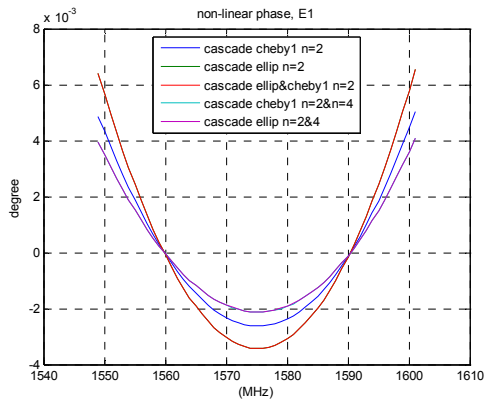
Figure 7.16c to *Figure 7.16f* shows the non linear phase in degrees at E1, E5, E6 and the combined band (from 0 Hz to 1595.88 MHz) respectively. In each plot, the non-linear phase distortion differs with different cascaded filters, but the overall scale is very small i.e. less than 0.008, 0.02 and 0.004 degree for E1, E5 and E6 respectively.



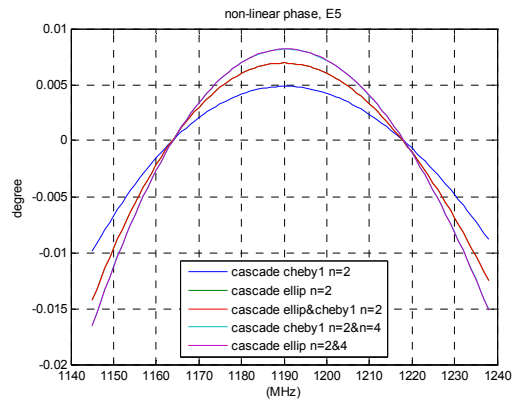
(a). Phase response up to 2 GHz



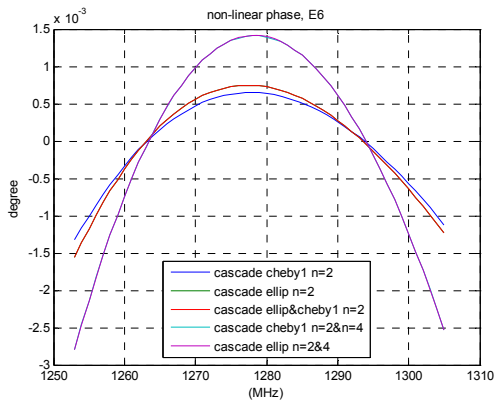
(b). Non-linear phase up to 2 GHz



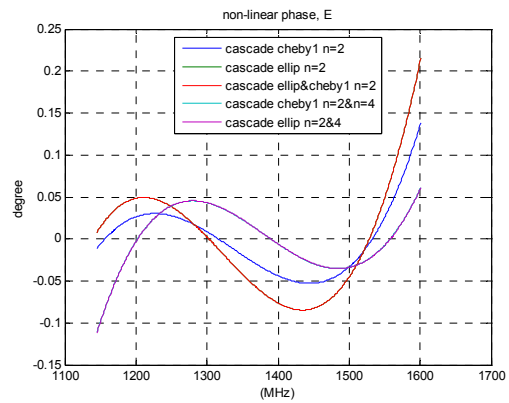
(c). Non-linear phase of E1 band



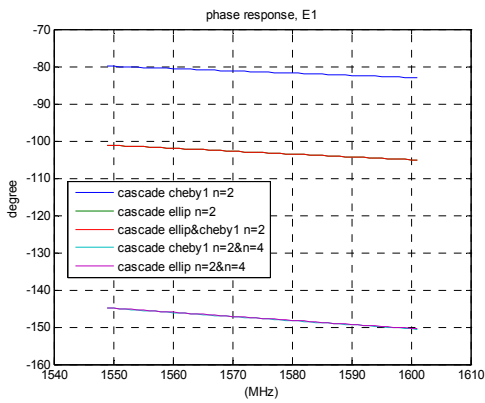
(d). Non-linear phase of E5 band



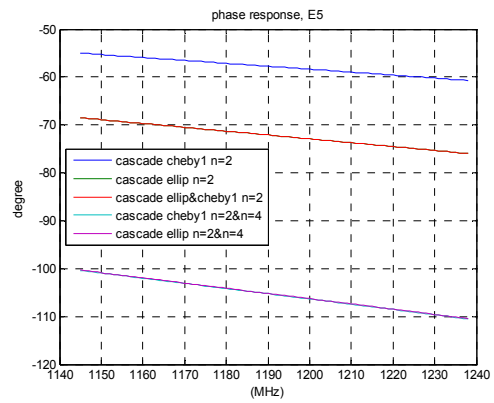
(e). Non-linear phase of E6 band



(f). Non-linear phase of whole band

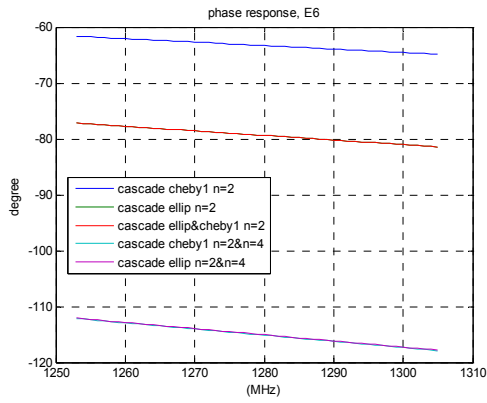


(g). The phase response of E1

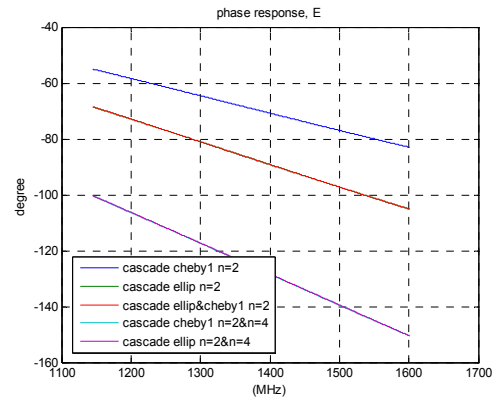


(h). The phase response of E5

Figure 7.16: Plots of phase response



(i). the phase response of E6



(j). The phase response of whole band

Mapping scheme for Figure 7.16:

- Blue..... 1.Cascaded Chebyshev I, order 2
- Green..... 2. Cascaded elliptic, order 2
- Red.....3.Cascaded elliptic and Chebyshev I, order 2
- Cyan.....4.Cascaded Chebyshev I, order 2 and 4
- Magenta.....5.Cascaded elliptic, order 2 and 4

Figure 7.16: Plots of phase response

7.3.4 Conclusion

From above analysis, the phase distortion in each interesting band is much smaller than the minimum industrial requirement for non-linear phase distortion i.e. 0.3 degree. Therefore the non-linear phase distortion of the NeptuneV5 card can be neglected for each navigation signal band. Figure 7.16g to Figure 7.16j shows the phase response of E1, E5, E6 and the combined band, respectively. As illustrated in these figures, the phase responses can also be considered as linear in this case.

7.4 SNR analysis for Down converter

To see how much the quantization degrades the performance a signal at the center frequency of each band has been tested. They have been process both by the FPGA implementation and by the floating point version of the MATLAB model. As can be seen by comparing *Table 7.2* and *Table 7.3* the worst degradation is 0.25 dB which is the case for the E6 data path. This is an acceptable result. To measure only the cross channel leakage the common spurs from the ADC and quantization noise would have to be removed. This has not been done but instead the cross channel leakage values in the tables contain all noise. This mean it is not possible to add up the noise numbers because that would count the quantization noise and the spurs from the ADC more than once so they should be considered as worst case values. The values in the tables measure the complete output bandwidth and not only the band of interest.

Table 7.2: Floating point down converter

| | | E1 | E5 | E6 | Unit |
|---------------|-----------|-----------|-----------|-----------|------------------|
| S | | 14389624 | 20819998 | 17605946 | LSB ² |
| | N | 380 | 601 | 405 | LSB ² |
| SNR | | 45.78 | 45.40 | 46.38 | dB |
| X-chan | E1 | N/A | 793 | 415 | LSB ² |
| | | N/A | 44.19 | 46.28 | dB |
| | E5 | 492 | N/A | 576 | LSB ² |
| | | 44.66 | N/A | 44.85 | dB |
| | E6 | 295 | 611 | N/A | LSB ² |
| | | 46.88 | 45.32 | N/A | dB |

Table 7.3: Quantized down converter

| | | E1 | E5 | E6 | Unit |
|---------------|-----------|-----------|-----------|-----------|------------------|
| S | | 14391301 | 20809751 | 17609088 | LSB ² |
| | N | 387 | 618 | 429 | LSB ² |
| SNR | | 45.69 | 45.27 | 46.13 | dB |
| X-chan | E1 | N/A | 857 | 425 | LSB ² |
| | | N/A | 43.85 | 46.17 | dB |
| | E5 | 496 | N/A | 610 | LSB ² |
| | | 44.63 | N/A | 44.6 | dB |
| | E6 | 344 | 617 | N/A | LSB ² |
| | | 46.22 | 45.28 | N/A | dB |

7.5 FPGA Budgets

This section presents the size of the VHDL implementation. In *Table 7.4* the size of DownConv is shown with the FPGA size as reference. In *Table 7.5* the size of the individual components is shown with the total size of the DownConv block as reference.

Table 7.4: Total size of down converter in FPGA

| Part | Slice registers | | Slice LUTs | |
|-------------------|-----------------|-----|------------|-----|
| | [No. of] | [%] | [No. of] | [%] |
| xc5vsx95t-1ff1136 | 58 880 | 100 | 58 880 | 100 |
| DownConv | 10 241 | 17 | 15 671 | 27 |

Table 7.5: Detailed size of individual down converter blocks

| Part | Slice registers | | Slice LUTs | |
|---------------------------------|-----------------|-------|------------|-------|
| | [No. of] | [%] | [No. of] | [%] |
| DownConv | 10 241 | 100,0 | 15 671 | 100,0 |
| First stage mixer | 41 | 0,4 | 20 | 0,1 |
| First stage FIR | 2 027 | 19,8 | 2 235 | 14,3 |
| First stage requant. | 49 | 0,5 | 196 | 1,3 |
| E1, second stage mixer | 1 145 | 11,2 | 1 740 | 11,1 |
| E1, second stage mixer requant. | 53 | 0,5 | 188 | 1,2 |
| E1, second stage FIR | 960 | 9,4 | 1 556 | 9,9 |
| E1, second stage FIR requant. | 29 | 0,3 | 106 | 0,7 |
| E5, second stage mixer | 1 342 | 13,1 | 2 035 | 13,0 |
| E5, second stage mixer requant. | 52 | 0,5 | 188 | 1,2 |
| E5, second stage FIR | 2 328 | 22,7 | 3 738 | 23,9 |
| E5, second stage FIR requant. | 29 | 0,3 | 112 | 0,7 |
| E6, second stage mixer | 1 145 | 11,2 | 1 745 | 11,1 |
| E6, second stage mixer requant. | 52 | 0,5 | 188 | 1,2 |
| E6, second stage FIR | 960 | 9,4 | 1 518 | 9,7 |
| E6, second stage FIR requant. | 29 | 0,3 | 106 | 0,7 |

8 CONCLUSION

This project has shown that it is possible to implement a Galileo down converter in FPGA using VHDL. To save hardware the design uses a cascade type design and re-quantization between the multiplication steps. The project shows that the re-quantization does not degrade the performance out of specification.

Verifying hardware is non trivial and time consuming but important. In this project a bit true MATLAB model is used as a shortcut to save time. A further improvement of the work can be to write VHDL test benches for the individual blocks used in the down converter.

When working only with digital logic the ADC is the limiting factor. This project has investigated the performance of the input stage of the ADC and shown that it is performing good enough to fulfill the specifications for the down converter.

The final conclusion is that the proposed architecture meets all requirements.

9 List of References

- [1] European Space Agency (2007) *What is Galileo*. [online] available from <http://www.esa.int/esaNA/GGGMX650NDC_galileo_0.html> [17 July 2007]
- [2] Ziemer,R. and Peterson,R. (2001) *Introduction to Digital Communication*. 2nd edn. New Jersey:Prentice Hall, 89-92
- [3] Proakis, J. G. and Manolakis, D.G. (2006) *Digital Signal Processing, Principles, Algorithms and Applications*. 4th edn. New Jersey:Prentice Hall
- [4] TEK Microsystems, Inc., Hardware Reference Manual Neptune V5 VXS
- [5] TEK Microsystems, Inc., Neptune-V5 VXS DK01 Application Developers Kit Manual
- [6] Maloberti,F. (2007) *Data converters*. Dordrecht: Springer.
- [7] Arpaia, P. et al. (2002) *A State of the Art on ADC Modeling*. [online] available from <<http://www.imeko.org/publications/tc4-2002/IMEKO-TC4-2002-014.pdf>> n.d.