UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DESIGN AND CONSTRUCTION OF A DIGITAL SIDEBAND SEPARATING SPECTROMETER FOR THE 1.2-METER SOUTHERN RADIO TELESCOPE

TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA

RICARDO ALBERTO FINGER CAMUS

PROFESOR GUÍA:
FAUSTO PATRICIO MENA MENA

MIEMBROS DE LA COMISIÓN:
LEONARDO BRONFMAN AGUILÓ
JAMES R. FISHER
MATTHEW A. MORGAN

Santiago de Chile, 2013.

Este trabajo ha sido parcialmente financiado por: CONICYT, a través de sus programas Fondo Basal, proyecto PBF06, FONDEQUIP, Fondo Gemini N° 32100003 y Fondecyt Project No. 1121051. Agradecemos a Xilinx Inc. Por la donación de circuitos integrados y licencias de software.

Abstract

Dual sideband (2SB) receivers are well suited for the spectral observation of complex astronomical signals over a wide frequency range. They are extensively used in radio astronomy, their main advantages being to avoid spectral confusion and to diminish effective system temperature by a factor two with respect to double sideband (DSB) receivers. Using available millimeter-wave analog technology, wideband 2SB receivers generally obtain sideband rejections ratios (SRR) of 10 to 15dB, insufficient for a number of astronomical applications.

In this work we present the architecture of typical astronomical dual sideband receivers and describe the main causes limiting the performance of current analog technology. We elaborate on the necessity of high sideband rejection for astronomical observations and propose a new approach based on the use of digital technology to overcome the main problems limiting the sideband rejection of current instruments.

During this work we studied digital technics to improve the performance of sideband separating receivers. We report the design and implementation of a Field Programmable Gate Array (FPGA) based sideband separating Fast Fourier Transform (FFT) spectrometer including the implementation of a 4 GHz analog front end built to test the designs and measure sideband rejection. The setup uses a 2SB front end architecture, except that the mixer outputs are directly digitized before the IF hybrid, using two 8bits ADCs sampling at 1GSPS. The IF hybrid is implemented on the FPGA together with a set of calibration vectors that, if properly chosen, compensate for the analog front-end amplitude and phase imbalances. The calibrated receiver exhibits a sideband rejection ratio in excess of 40 dB for the entire 2 GHz RF bandwidth, which represent and improvement of a factor 100 to 1000 respect to current radio astronomy receivers.

Resumen

Los receptores de doble banda lateral (2SB) son particularmente útiles para la observación de espectros astronómicos complejos en un amplio rango de frecuencias. Son extensamente utilizados en radio astronomía siendo sus principales ventajas el evitar la confusión espectral y disminuir la temperatura efectiva de sistema en un factor de dos con respecto a los receptores de doble banda lateral (DSB). Usando la actual tecnología analógica, los receptores 2SB de banda ancha obtienen generalmente cocientes de rechazo de banda lateral (SRR) de 10 a 15 dB, valores insuficientes para algunas aplicaciones astronómicas.

En este trabajo se presenta la arquitectura típica de los receptores astronómicos de doble banda lateral y se describen las principales causas que limitan el rendimiento de la tecnología analógica actual. Se elabora sobre la necesidad de un alto rechazo de banda lateral para observaciones astronómicas y se propone un nuevo enfoque usando tecnología digital para superar los problemas que limitan el rechazo de banda lateral de los instrumentos actuales.

Durante este trabajo se estudiaron técnicas digitales para mejorar el rendimiento de los receptores con separación de banda lateral. Se presenta el diseño e implementación de un espectrómetro de transformada de Fourier rápida (FFT) con separación de banda lateral digital incluyendo la implementación de un receptor analógico de 4 GHz construido para probar los diseños y medir el rechazo de banda lateral. La configuración utiliza una arquitectura clásica de receptor 2SB, excepto que las salidas de los mezcladores son directamente digitalizadas, antes del híbrido de IF, utilizando dos ADCs de 8 bits a 1 GSPS. El híbrido de IF está implementado en la FPGA junto con un conjunto de vectores de calibración que, debidamente elegidos, compensan los desequilibrios de amplitud y fase del receptor analógico. El receptor calibrado exhibe un cociente de rechazo banda lateral superior a 40 dB para todo el ancho de banda de recepción de 2 GHz. Esto representa una mejora de un factor de 100 a 1000 respecto a los actuales receptores radio astronómicos.

We remember professor Jorge May (R.I.P.) for his support to this project. I would also like to thank the complete team at the Millimeter Wave Laboratory for their invaluable trust and support.

Contents

Table of Figures

1. Introduction

1.1. General concepts

Most receivers used in radio astronomy are similar to those used in telecommunication. They are called heterodyne or coherent receivers. The main feature of these receivers is that they convert the incoming signal to a lower frequency range conserving the phase and amplitude information of the original. This low frequency signal, known as Intermediate Frequency (IF), is processed by the back-end electronics to extract the information needed for a particular observation. Nevertheless the principle of operation of a radio astronomy and telecommunication receiver is the same, the former has normally better performance in terms of bandwidth and noise. The simplest configuration of a double side band (DSB) receiver is shown in Figure 1.

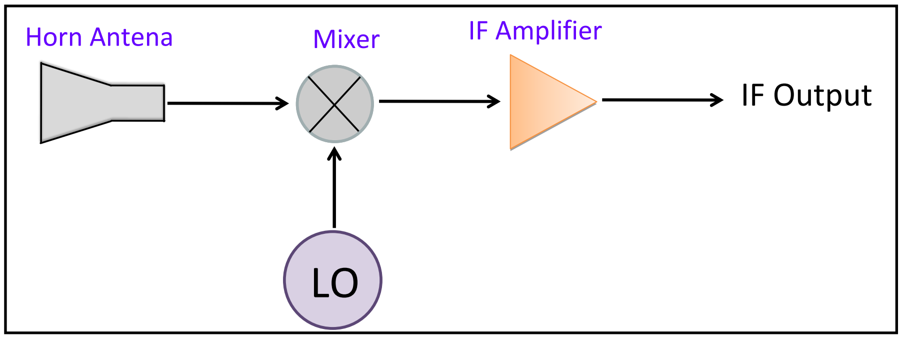


Figure 1: Basic block diagram of a DSB receiver. For high frequency receivers SIS mixers are preferred as a front end.

In all heterodyne receivers a nonlinear device, called a mixer, processes the incoming signal and a reference signal known as the Local Oscillator (LO). As a result of the nonlinearity, mixing products are generated that can be harmonics or inter-modulation products. A low pass filter normally selects the signal that has a frequency equal to the difference in frequency between LO and RF, a process called downconversion.

The main problem with this strategy is that frequencies above the LO (upper sideband or USB) and below the LO (lower sideband or LSB) are downconverted to the same IF frequency as pictured in Figure 2. (Thompson, 1986)

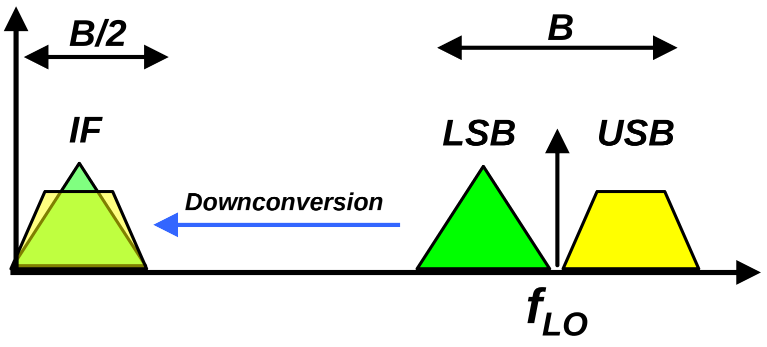


Figure 2: Sideband overlapping on a DSB receiver. In this example the LO frequency f_{LO} is at the center of the RF spectrum of bandwidth B. After downconversion the USB and LSB are overlapped at the IF.

One approach to overcome the sideband overlapping is to filter one sideband before injecting the signal into the mixer. This kind of receiver is called a single sideband receiver (SSB). SSB receivers are practical when the LO is fixed or when the IF band is away from zero and the RF bandwidth is comparable with the IF bandwidth, like ALMA Band 1 (Reyes, 2013). Even though for some applications SSB receivers are suitable, they are not practical for most high frequency, tunable, wide bandwidth applications. For example ALMA bands 3 to 10 use either sideband separating mixers or double side band mixers (ALMA, 2013).

Sideband Separating Mixers (SSM) are designed to overcome this problem separating the USB and LSB in two different outputs. Figure 3 shows the classical configuration of a dual sideband separating mixer. This configuration includes two mixers and two quadrature hybrids at the RF and IF bands.

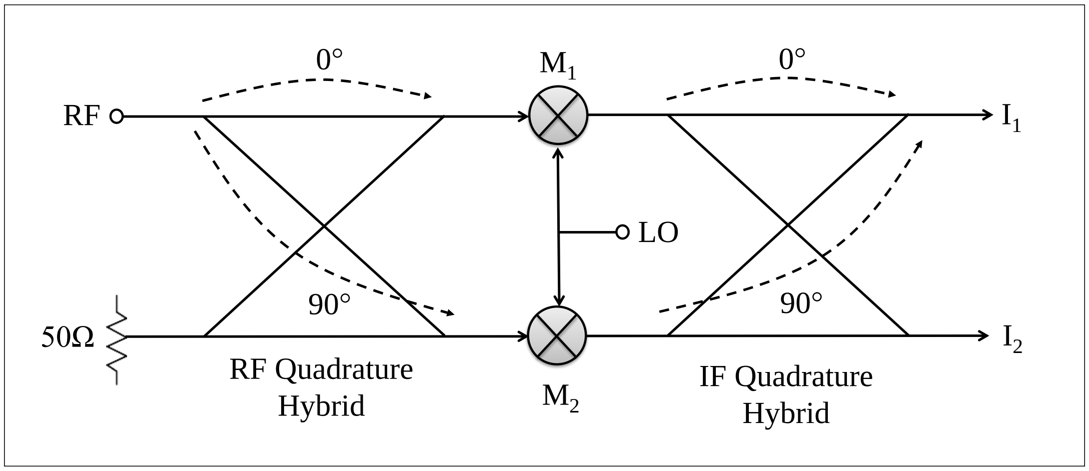


Figure 3: Basic Sideband Separation Mixer (SSM) configuration.

A Quadrature Hybrid is four-port device that produces two outputs by phase shifting and adding the two inputs. There are many physical realizations of this device depending mainly on the frequency of operation. These realizations go from pure geometrical waveguide or microstrip designs to fully-electronic OP-AMP based designs. The realization of any quadrature hybrid can be described by the model shown in Figure 4. This model is a good representation of the actual behavior of hybrids only for small bandwidths. Far away from the central design frequency all hybrids deviate from quadrature (90° shift) and begin to show amplitude imbalances.

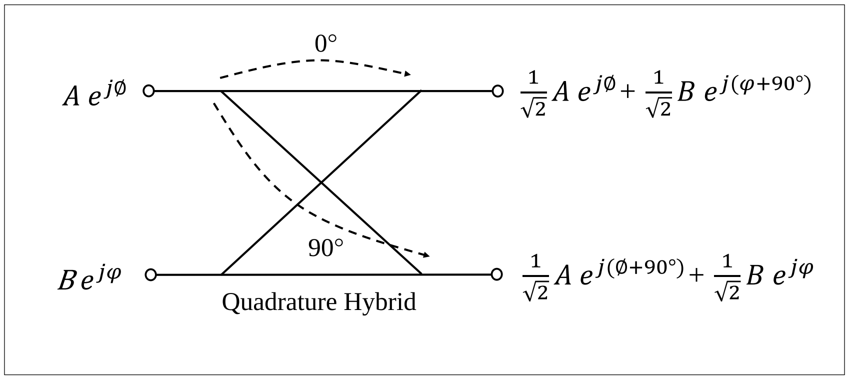

0°
$A\,e^{j\emptyset}$
$\frac{1}{\sqrt{2}}A\,e^{j\emptyset}+\frac{1}{\sqrt{2}}B\,e^{j(\varphi+90°)}$
90°
$B\,e^{j\varphi}$
$\frac{1}{\sqrt{2}}A\,e^{j(\emptyset+90°)}+\frac{1}{\sqrt{2}}B\,e^{j\varphi}$
Quadrature Hybrid

Figure 4: Representation of an Ideal Quadrature Hybrid Model.

Referring back to Figure 3, in a 2SB receiver the RF hybrid outputs are downconverted by the mixers M1 and M2, which are driven by the same LO, and further recombined by the IF hybrid producing the outputs I1, I2. It is easy to show (Bert C. Henderson, 1985) that the outputs I1, I2 are the USB and LSB components of the incoming RF signal. Actual SSMs mixers include other elements like amplifiers, isolators and filters in addition to the components shown in Figure 3. However, current analog technology is limited in the degree to which it can separate the two sidebands.

The Sideband (or “Image”) Rejection Ratio (SRR) is one of the most important figures of merit of a sideband separating mixers, either single sideband (SSB) or dual side band (2SB) mixers. It is defined as the power ratio of the wanted sideband to the unwanted (rejected) sideband on each IF output. Ideally, the SRR should be infinite for all frequencies on both outputs of a 2SB mixer. In practice, unavoidable gain and phase imbalances strongly limit the achievable sideband rejection ratio to about 10-20 dB for purely high sensitivity, broadband receivers. The difficulty of getting a high separation ratio is due to the challenge of producing two parallel RF/IF channels (including an RF hybrid, mixers, amplifiers, filters, and IF hybrid) with excellent amplitude and phase balance over broad bandwidths. We present here the design and implementation of a digital sideband separating spectrometer which makes use of the latest advancements in digital technology to improve the performance of sideband separating receivers.

1.2. The State of the Art

Much effort has been invested during the last decades to improve the sensitivity of radio telescopes. Cryogenic heterodyne receivers are quickly approaching fundamental limits to their noise temperature, but they are showing a relatively modest performance in other figures like Sideband Rejection Ratio.

To estimate the sideband rejection ratio we have to take into account the amplitude imbalance, A, equal to the sum of the individual imbalanced in the RF and IF path as well as the value, θ, defined as the total phase imbalance due to the quadrature deviations on the hybrids and any other imbalance in the electrical length of both signal paths. In (Bert C. Henderson, 1985) it is shown that when A and θ are included in the 2SB mixer model we can calculate the Image Rejection (SRR) as follow:

$$SRR = -10 \log \left(\frac{1 + A^2 - 2A \cos\theta}{1 + A^2 + 2A \cos\theta} \right)$$

$$A = 10^{-\left(\frac{A\,dB}{20} \right)}$$

Note that for A=0 dB and θ=0° SRR becomes (+) infinite which is the ideal case. The amplitude imbalances are not only produced by asymmetries *within* hybrids and mixers but also due to impedance mismatches *between* the different components in the system, including amplifiers, isolators and filters before the IF hybrid. Figure 5 shows how the sideband rejection ratio decreases when amplitude and phase imbalances are present.

In Figure 5 it is easy to see how difficult it is to get sideband rejection ratios above 20 dB. To achieve this goal it is necessary to build two RF paths with an amplitude imbalance less than 1.5dB and a phase imbalance less than 5 degrees over the entire receiver bandwidth.

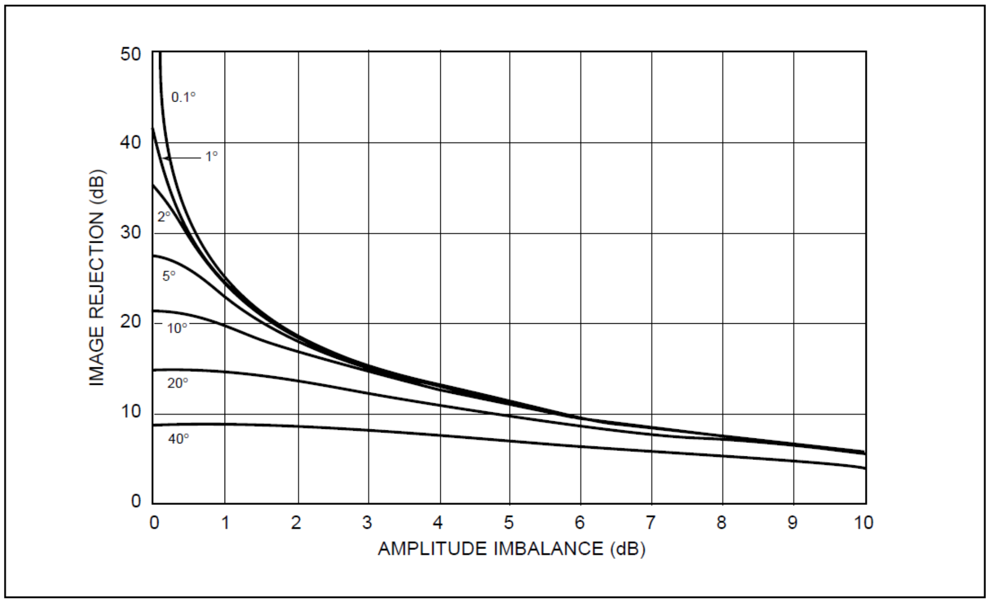


Figure 5: Sideband Rejection Ratio for different phase and amplitude imbalances. From (Bert C. Henderson, 1985).

The Atacama Large Millimeter/submillimeter Array (ALMA) specification for its state-of-the-art receivers requires an SRR better than 7dB over the entire band and better than 10dB over the 90% of the band (Cunningham *et al*. 2007). What seems to be a moderate number proved to be difficult to achieve, particularly at the higher frequencies (F. P. Mena, 2011) (Mahieu, 2011). Figure 6 shows the typical performance of an ALMA band-3 receiver (Ma, 2011), which covers a similar frequency range as the southern mm-wave radio telescope.

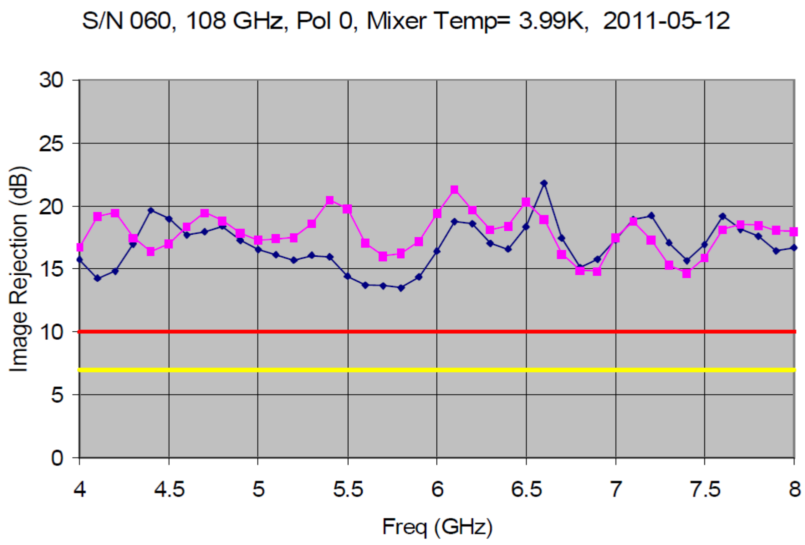


Figure 6: Sideband rejection ratio of an ALMA band 3 cryogenic receiver. The red and yellow lines represent the ALMA specifications for 90% and 100% of the band, respectively. Blue line is USB and violet line is LSB. From (Ma, 2011).

1.3. On the necessity of high sideband rejection ratio

The overlap of the upper and lower sideband spectra complicate observations in a number of ways. Some important issues are described below.

1.3.1. Noise reduction

In all radio telescopes the signal collected by the antenna is a sum of the astronomical signal and noise coming from several sources such as microwave background, atmospheric emission, antenna spillover and optics (lenses and filters). A low sideband rejection ratio makes the noise of both sidebands to be present in the IF output, therefore increasing the system noise temperature. When the atmospheric emission is high or when the noise contribution from the optics is intense, a high sideband rejection ratio becomes particularly important. Good sideband rejection also allows better calibration of atmospheric emission in single dish observation. (Lucas, 2000) (D'Addario, 2000)

1.3.2. Observation efficiency

A technique to cope with low sideband rejection instruments makes use of the fact that the frequency of the signals coming from USB and LSB shifts in opposite directions within the IF band when the LO is displaced. To use this technique at least two observations of the same source with different LO settings are necessary. This produces a reduction of observation efficiency by at least a factor of 1/2 that is critical when observing very faint sources. (Vilaro, 2011)

1.3.3. Determine complex sources

When the radio source is too complex, i.e. made of many spectral lines, broad lines, present at both sidebands, it can be difficult to untangle in the IF spectrum even when multiple observations with different LOs are available. In this case a high sideband rejection receiver is essential. (Mangum, 1998)

1.3.4. Reduction of radio frequency interference (RFI)

With the extensive use of the electromagnetic spectrum for telecommunications and radar/positioning systems it has become frequent to make astronomical observations close to satellite or ground-based commercial bands. This is particularly true for relatively low frequency radio telescopes like the SKA (R.P. Millenaar, 2011). The received power from man-made transmissions is normally many orders of magnitude above astronomical sources. If these transmissions are present in the image band for an astronomical observation, a high sideband rejection will be critical. For example, in the band of our interest there are radar applications at 94GHz (Heymsfield, 2008) (A. Surkov, 2006), and even applications above 100GHz have been proposed (Wallace, 2010).

2. A new approach

The increasing power of digital processing hardware has opened the door for a new approach which is based on the idea of performing the IF recombination (IF hybrid, Figure 3) in digital signal processing. This method allows correction of the imbalances of the analog components with digital processes.

In fact, a receiver with digital sideband separation was reported on 2009 (Axel Murk, 2009). For the back end of this receiver they made use of a commercially available digital signal analyzer (AC240 from Acqiris/Agilent) capable of digitizing 2×500MHz of bandwidth, and an on-board Field Programmable Gate Array for high resolution and broadband signal processing in real-time.

With this digital back end they were able to digitize the mixers outputs of a 340 GHz front end and build (in digital) an ideal IF hybrid for sideband separation. They reported image rejections between 10 and 25 dB, which are at the high end of current analog separating technology. Even though they did not use the digital back end to correct for imbalances in the RF front end, they clearly showed the applicability of FPGA based platforms to do signal processes normally performed by analog hardware.

Another interesting example of digital processing is the conversion from linear to circular polarization using FPGA (Koyel Das, 2011). In this work a 512 MHz bandwidth was processed by a logic simulator to form circular polarization by quadrature phase shifting and summing the equalized linear polarization signals. Polarization purity of -25 dB was obtained. Even though the technique was demonstrated using a logic simulator, the implementation on a real FPGA is relatively straightforward. The authors consider that the trend in next-generation receivers for radio astronomy is to move the samplers as close as possible to the front end. (Koyel Das, 2011)

A prototype proving calibrated sideband separation has been recently (2010) constructed by M.A. Morgan and J.R. Fisher at NRAO. In their design, Morgan and Fisher processed 250 MHz of bandwidth downconverted from L-Band (1.2-1.7GHz) with a sideband rejection ratio better than 50dB over the entire band. This represents an outstanding sideband separation 20 to 30 dB better than current analog technology (Morgan and Fisher, 2010).

The method consists of digitizing the mixer outputs to then calculate the Fourier transform. After the Fourier transform is calculated, each frequency channel (from both mixers) is duplicated and multiplied by a complex constants C_i, i=1,...,4. Figure 7 shows the architecture of Morgan and Fisher's Digital Sideband Separating Mixer (DSSM) (Fisher and Morgan, 2008).

C_1 to C_4 represent 1-dimesional complex arrays, of a length equal to the number of channels of the FFT. When C_1 to C_4 are chosen carefully one can not only reproduce digitally the behavior of an ideal IF hybrid but calibrate out the accumulated imbalances of both signal branches for each frequency channel. It has been shown that this scheme dramatically increases the sideband rejection ratio of sideband separating receivers (Morgan and Fisher, 2010).

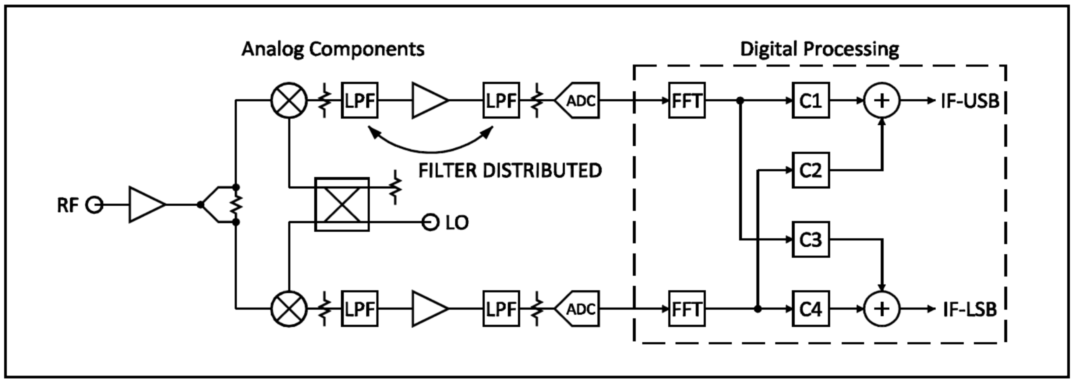


Figure 7: Morgan and Fisher Digital Sideband Separating Mixer (DSSM). From (J. R. Fisher, 2008).

Figure 8 shows the sideband rejection achieved by M.A. Morgan and J.R. Fisher after calibration was performed. Figure 8(a) shows the image rejection when the receiver was at the same temperature at which the calibration was performed (28°C). Figure 8 shows the sideband rejection when the temperature of the receiver was increased up to 40°C. Even though the digital processing performed in this experiment was off-line on a desktop computer, it does show the possibility of increasing the sideband rejection ratio of current receivers by 20 to 30dB. It also shows that the calibration is stable enough under temperature variations.

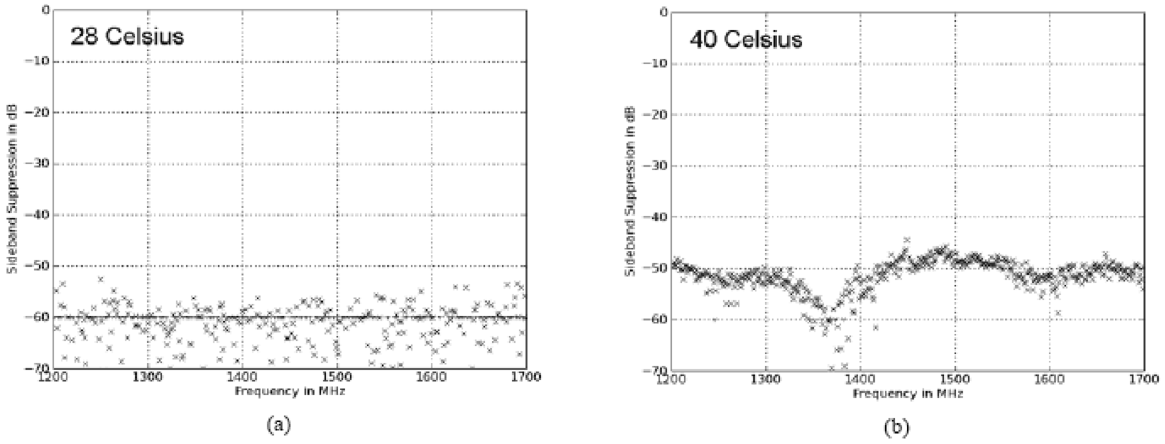


Figure 8: Sideband isolation measured in the lab at (a) 28°C (b) 40°C. Points below the noise of the test set were conservatively marked at -60dB as an upper bound. Calibration was performed at 28°C. From (Fisher and Morgan, 2008).

3. Spectrometer Design

3.1. Hardware

Figure 9 shows the block diagram of the sideband separating spectrometer that is the subject of this thesis.

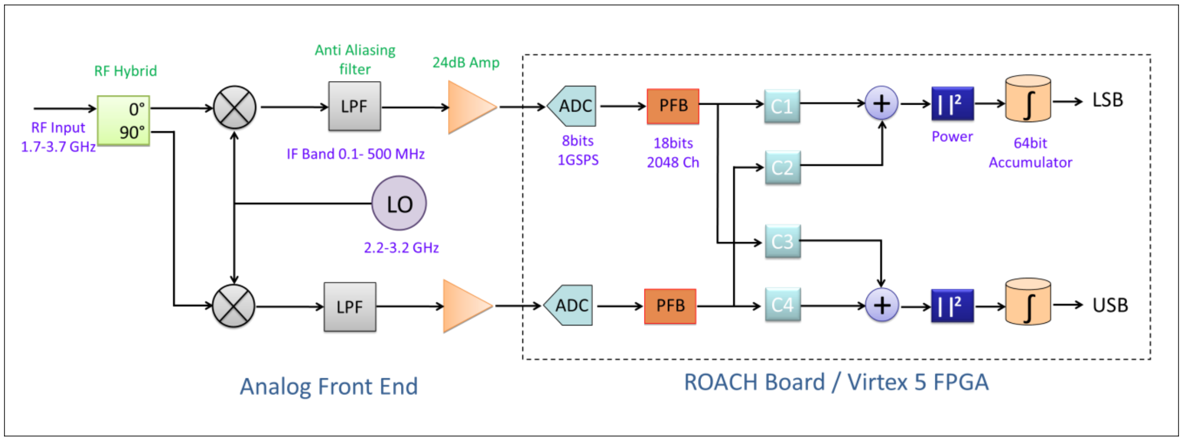


Figure 9: Digital Sideband Separating Receiver Block Diagram. The front end was built out of commercial parts to provide the functionality of typical analog receivers. The boundary between front end and back end is defined at the ADCs inputs in the ROACH assembly.

3.1.1. Front End

A 4 GHz analog front end was built out of commercial parts to provide the functionality of typical analog receivers. Following the 90° RF hybrid two mixers down-convert the input signal using a 2.2 to 3.2 GHz LO. Figure 10 shows a picture of the analog plate.

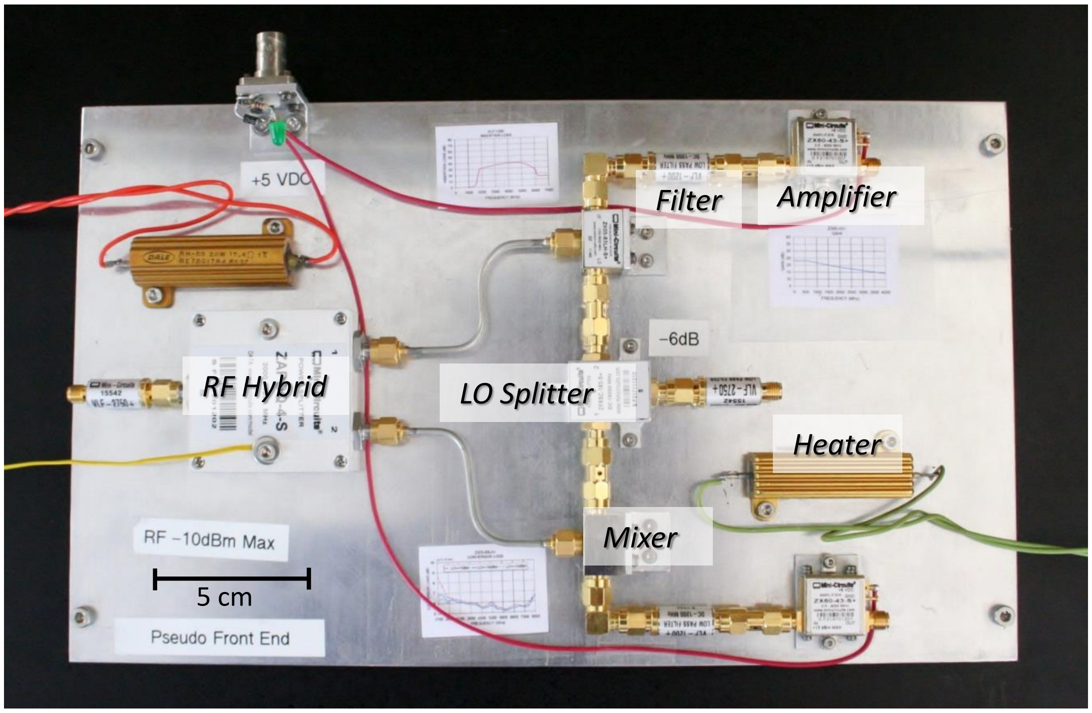


Figure 10: Front end analog plate. From left to right the RF input filter, RF quadrature hybrid, two semi-rigid cables leading to the mixers, anti-aliasing filters and amplifiers can be seen. In the center of the picture the LO input filter and LO splitter are seen. The filters are used to attenuate harmonics coming from the synthesizers. The gold corrugated components on the upper left and lower right sides are the heaters.

3.1.2. Back End

After amplification and anti-aliasing filtering the outputs are digitized to 8 bits, at 1 GSPS, allowing the processing of a 500 MHz IF bandwidth per sideband. Even though the RF frequency of this prototype is relatively low, there is nothing preventing the design to operate at higher RF frequencies provided a suitable front end.

Our design closely resembles the prototype of Morgan and Fisher but includes a power block and a 64bit accumulator to integrate the high data rate output within the FPGA. A simpler filtering scheme and in-phase LO injection were used in our front end. The complex constants C1 and C4 were set to 1, while C2 and C3 where adjusted to calibrate the phase and amplitude imbalances. C2 and C3 as well as the accumulation length are accessible to the user and can be modified from the control computer.

The hardware used to perform the signal processing is known as the Reconfigurable Open Architecture Computing Hardware (ROACH)[1]. The ROACH is an open FPGA-based (Xilinx Virtex 5) platform, the product of an international collaboration lead by the Center for Astronomy Signal Processing and Electronic Research (CASPER) in University of California at Berkeley. CASPER aims to produce open hardware designs and software/gateware resources for signal processing in astronomy[2].

Figure 11 shows the block diagram of the ROACH board. The centrepiece of the ROACH is a Xilinx Virtex 5 FPGA (either LX110T for logic-intensive applications, or SX95T for DSP-slice-intensive applications). A separate PowerPC chip runs Linux and is used to control the board, i.e., program the FPGA and allow interfacing between the FPGA software registers/BRAMs/FIFOs and external devices using Ethernet. The high performance Virtex-5 FPGA in which the ROACH is based can be programmed to perform very demanding parallel data processing algorithms usually found in radio astronomy.

Two quad data rate (QDR) SRAMs provide high-speed, medium-capacity memory, and one DDR2 DIMM provides slower-speed, high-capacity buffer memory for the FPGA. The PowerPC has an independent DDR2 DIMM to run the processes and the especially designed Linux distribution called "Berkeley Operating system for ReProgrammable Hardware" (BORPH)[3].

BORPH is an Operating System designed for FPGA-based reconfigurable computers. It is an extended version of the Linux kernel that handles FPGAs as if they were CPUs. BORPH introduces the concept of a 'hardware process', which is a hardware design that runs on an FPGA but behaves just like a normal user program. The BORPH kernel provides standard system

[1] http://casper.berkeley.edu/wiki/ROACH
[2] https://casper.berkeley.edu/
[3] http://bee2.eecs.berkeley.edu/wiki/Bee2OperatingSystem.html

services, such as file system access to hardware processes, allowing them to communicate with the rest of the system easily and systematically[4].

The two Z-DOK connectors allow ADCs, DACs and other interface cards to be attached to the FPGA. Four CX4 connectors provide a total of 40 Gbits/sec bandwidth for connecting ROACH boards together, or connecting them to other XAUI/10GbE-capable devices (such as computers with 10 GbE NICs and 10 GbE switches).

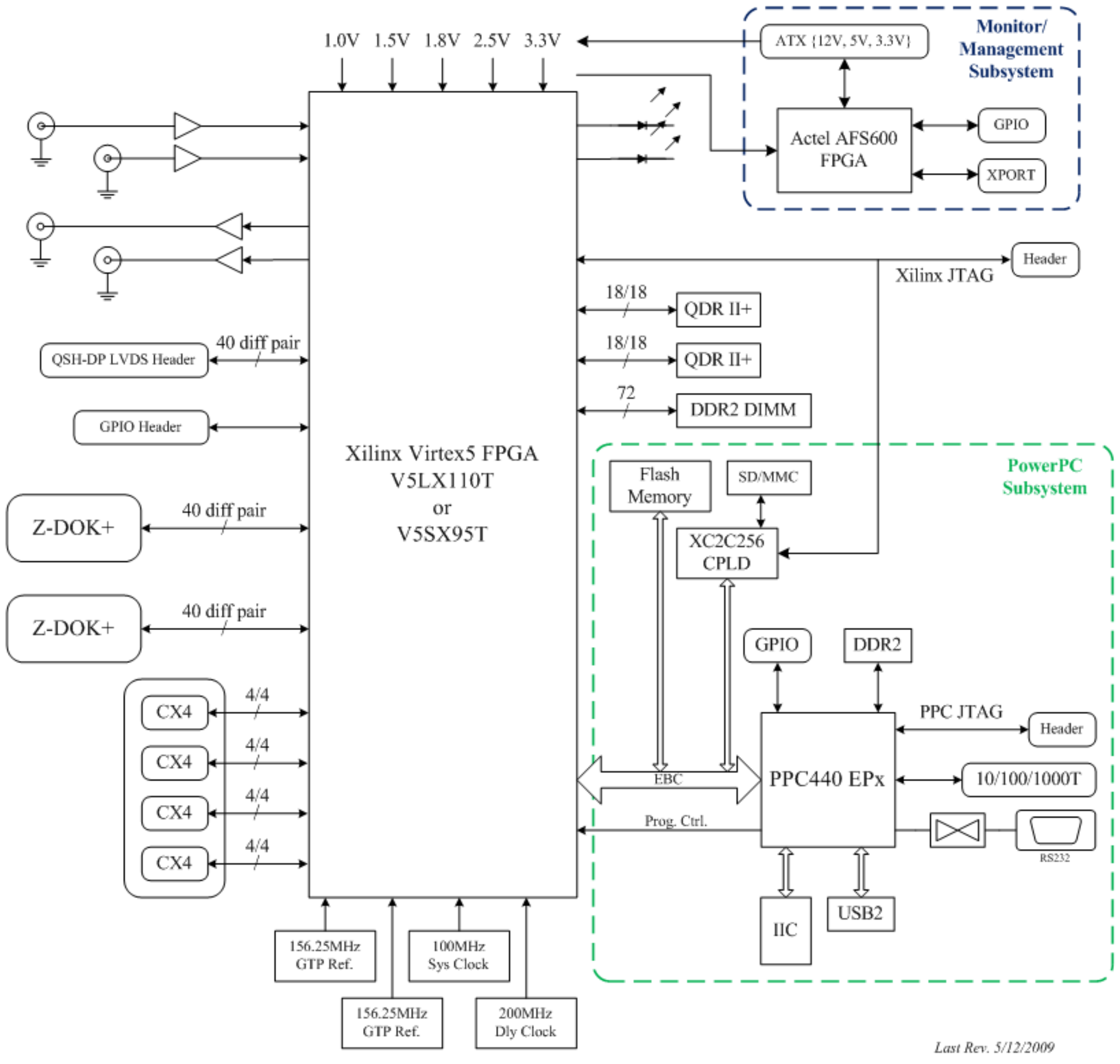


Figure 11: ROACH block diagram[1]. See the text for description.

The reason for choosing this platform is based on the flexibility of the design, the availability of documentation and the experience of use in astronomical data reduction. Leading institutions in radio astronomy like the National Radio Astronomy Observatory (NRAO) are collaborating with the design and development of this open hardware. Figure 12 shows a picture of the roach board in a typical configuration with two Analog to Digital Converters (ADCs) plugged into the Z-

[4] https://casper.berkeley.edu/wiki/BORPH

DOK ports. The ROACH platform allows the interconnection of multiple boards if it is necessary to grow in number of channels or computing power. A wiki page and mail list provides insights in other projects and experiences with the hardware, representing a rich source of information to plan, discuss and troubleshoot particular designs.

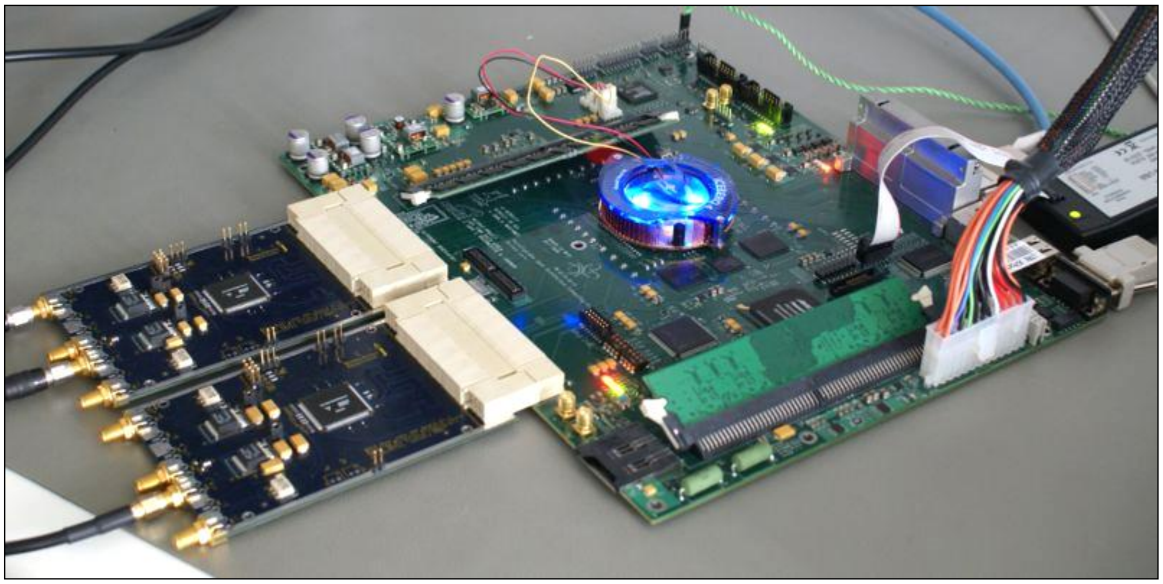


Figure 12: ROACH Board in a typical configuration.

The ADC board is based on the National Semiconductor ADC083000 chip[5] which is built using two interleaved 1.5 GSPS ADC cores driven by the rising and falling edge of the input clock. Figure 13 shows its internal configuration. The reference voltage (Vref in Figure 13) provided to each core as well as the gain of the input differential amplifiers (S/H in Figure 13) set the zero response of the ADC083000 chip. Even very small differences in these reference voltages/gains will appear -after interleaving- as a high frequency “ghost” periodic wave. This coherent tone falls above the first Nyquist zone and is, therefore, aliased down to the detected spectra. This and other spurious signal limits the spurious-free dynamic range to 50 dBc, when the ADC is driven at full scale, i.e. a test tone of 0 dBm. The maximum clock rate is 1.5 GHz, input sine wave. The clock is divided down by 4 before going into the FPGA. Consequently, the data is also demuxed by 4, resulting in 8 simultaneous samples.

[5] http://www.national.com/pf/DC/ADC083000.html#Overview

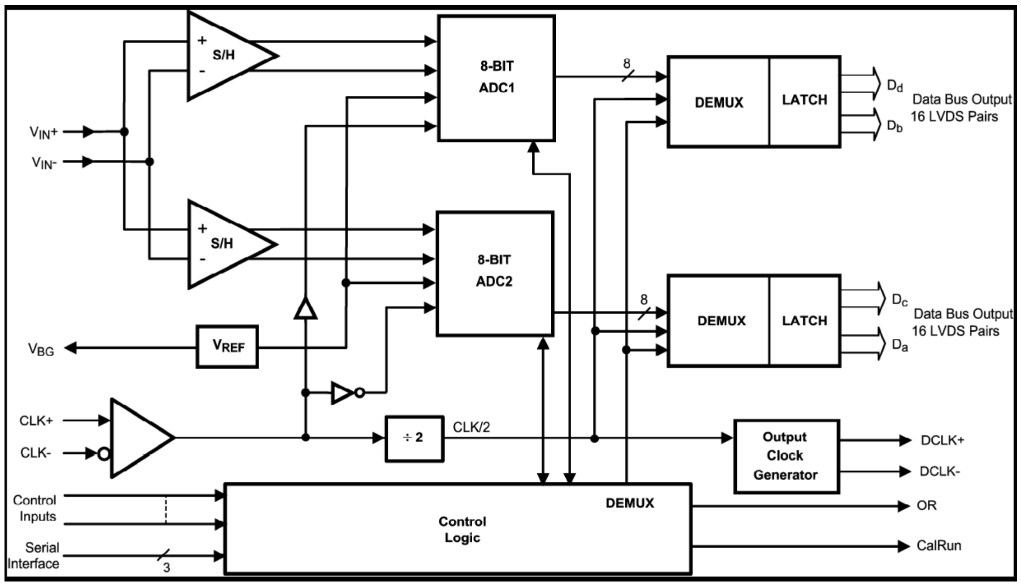


Figure 13: Internal block diagram of the National Instrument ADC083000 chip.

3.2. Gateware

Gateware is the representation of the data processing algorithm that runs on the FPGA. Programing an FPGA implies building hardware through selecting a particular interconnection of the hardware resources (e.g. logic gates, Digital Signal Processor (DSP) slices or memories). In contrast with normal computers that can only perform a small number of operations simultaneously, a FPGA can perform highly parallel processes including many operations on each clock cycle.

Figure 14 shows the high level design of the sideband separating spectrometer. Our design is based on the spectrometer example of CASPER tutorial 3, but includes a number of modifications.

The leftmost yellow block (adc083000x2) represents the two ADCs which are connected to two parallel spectrometers. The core element of each spectrometer is the Polyphase Filter Bank (PFB) block, which is made out of a Finite Impulse Response (FIR) filter followed by a pipeline Fast Fourier Transform (FFT), both symbolized by the green blocks on the model. The pipeline FFT is an algorithm which can compute the FFT in a sequential manner. It achieves real-time behavior with nonstop processing when data is continually fed through the processor (Bin Zhou, 2009).

Vector Complex Multipliers (VCM, blue blocks) were designed and incorporated before the power blocks. The VCM together with the DSP-based complex adders (a_add_dsp, white blocks) multiplies and adds the signal from both branches. A DSP-based power block (power_dsp_48e, in green) was used after combining the signals followed by a gain-requantization block mainly used for amplitude control and data type converting. Vectors Accumulators (VACC) integrate the power spectrum.

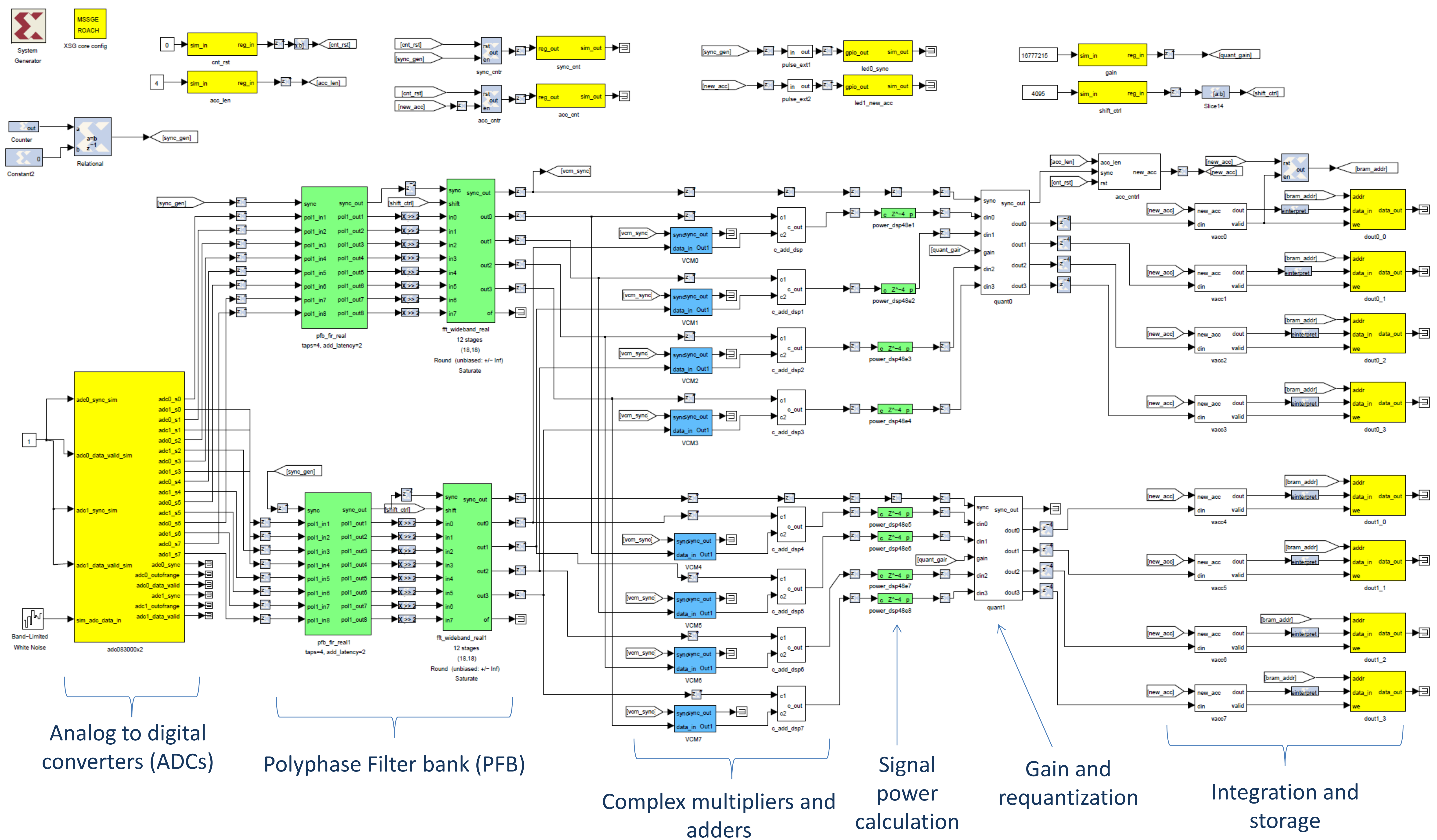


Figure 14: High level block diagram of the digital sideband separating spectrometer.

To prevent overflow and ease the need for requantization the vector accumulator depth was set to 64bit, and also 64bit Block Random Access Memories (BRAM) were used to store the data.

A FIR filter is a type of filter whose response to a finite input has a finite duration, i.e. the output $y[n]$ settles to zero in finite time after the input $x[n]$ is set to zero. A FIR filter output $y[n]$, is the weighted sum of the current and past inputs, $x[n]$ [6]. The FIR generic block diagram is shown in Figure 15.

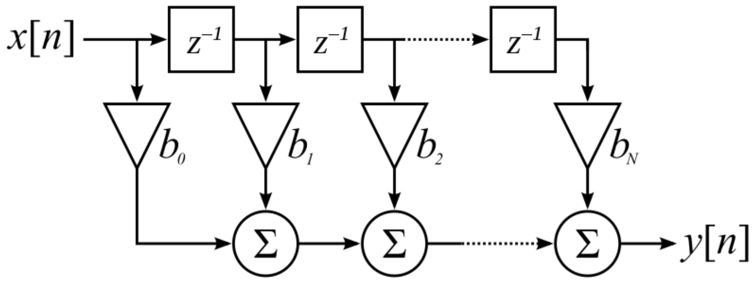
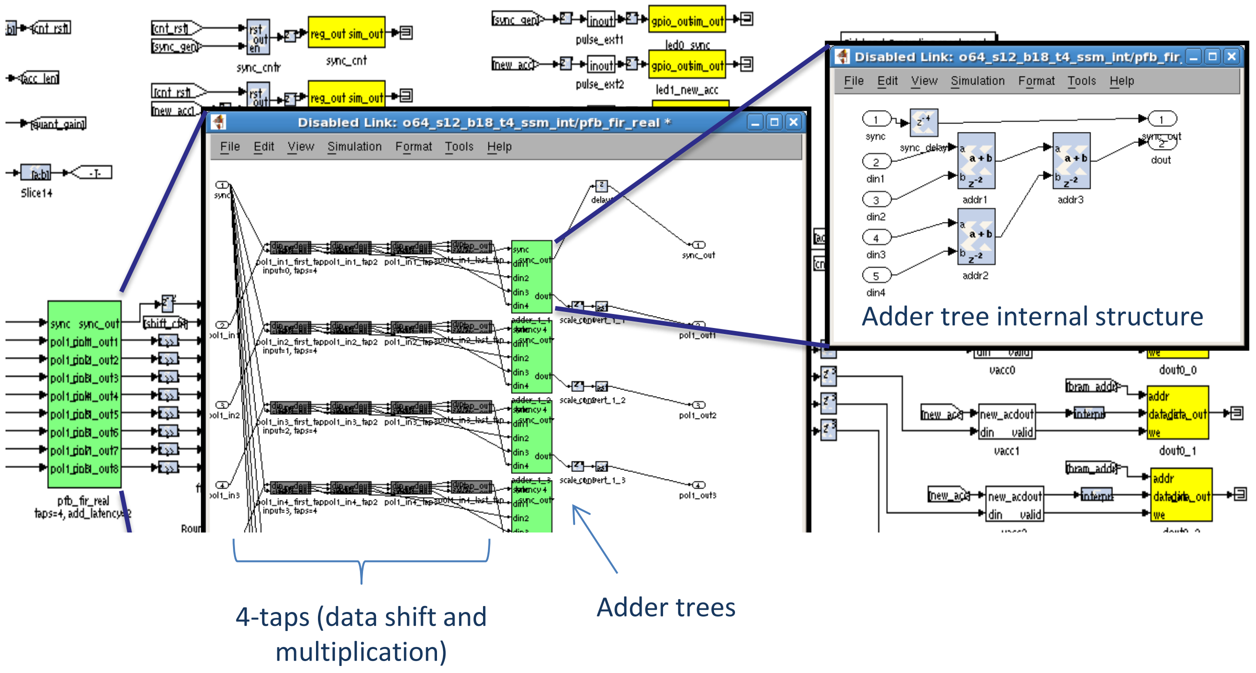


Figure 15: Finite impulse response filter bloc diagram. [7]

The order of the filter is the number of time samples that are used to calculate the output. A 4th order filter was used in our design. Figure 16 shows a zoom into the FPGA implementation.



Disabled Link: o64_s12_b18_t4_ssm_int/pfb_fir_real *
Disabled Link: o64_s12_b18_t4_ssm_int/pfb_fir_
File Edit View Simulation Format Tools Help
pfb_fir_real
taps=4, add_latency=2
sync
din1
din2
din3
din4
dout
sync_out
addr1
addr2
addr3
Adder tree internal structure
4-taps (data shift and multiplication)
Adder trees

Figure 16: 4th order finite impulse response filter implementation.

[6] Cambridge University web site: http://svr-www.eng.cam.ac.uk/~ajr/SA95/node13.html
[7] Figure taken from http://en.wikipedia.org/wiki/Finite_impulse_response

The addition of the FIR before the FFT improves the frequency response of each channel making them flatter within each frequency bin, sharper on the edges, and strongly reducing the inter-channel leaking. Figure 17 shows a comparison of the single-bin frequency response of a PFB with a direct FFT.[8]

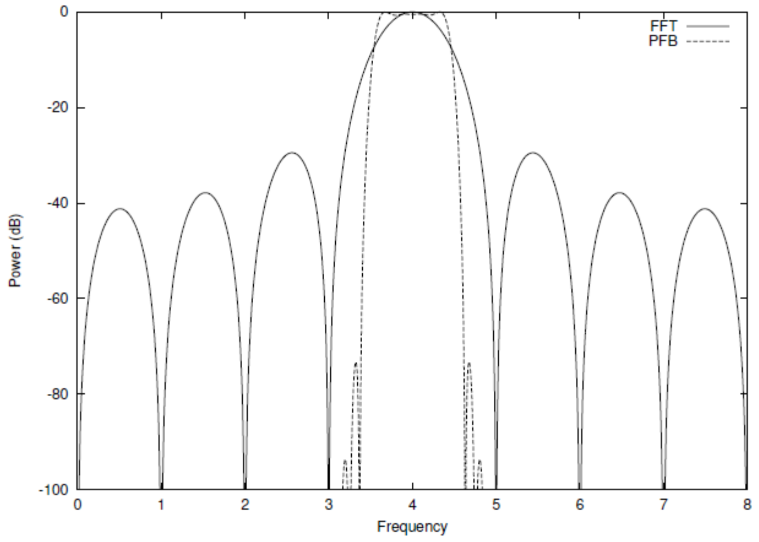


Figure 17: Comparison of the single-bin frequency response of a 4th order PFB with a direct FFT. [8]

The PFB built for the sideband separating spectrometer has 2048 channels with a data and coefficients bitwidth of 18 bits. Running at 1 GSPS, the resulting spectral resolution is 244 kHz, which is good for the observation of extended molecular clouds planned for the 1.2m mm-wave telescope. The coefficient and data bitwidth defines the level of numeric noise added by the PFB to the original signal. With 18bits the PFB noise floor is below -60 dBc when a full scale test tone is applied to the ADC input. The ADC spurious free dynamic range is 50 dBc, so the dynamic range of the spectrometer is limited by the ADC performance and not by the PFB noise.

The vector complex multipliers were designed to be based on a pair of BRAM memories for the real and imaginary coefficients and a complex DSP-based multiplier. The block diagram of the VCM is shown in Figure 18.

[8] https://casper.berkeley.edu/wiki/The_Polyphase_Filter_Bank_Technique

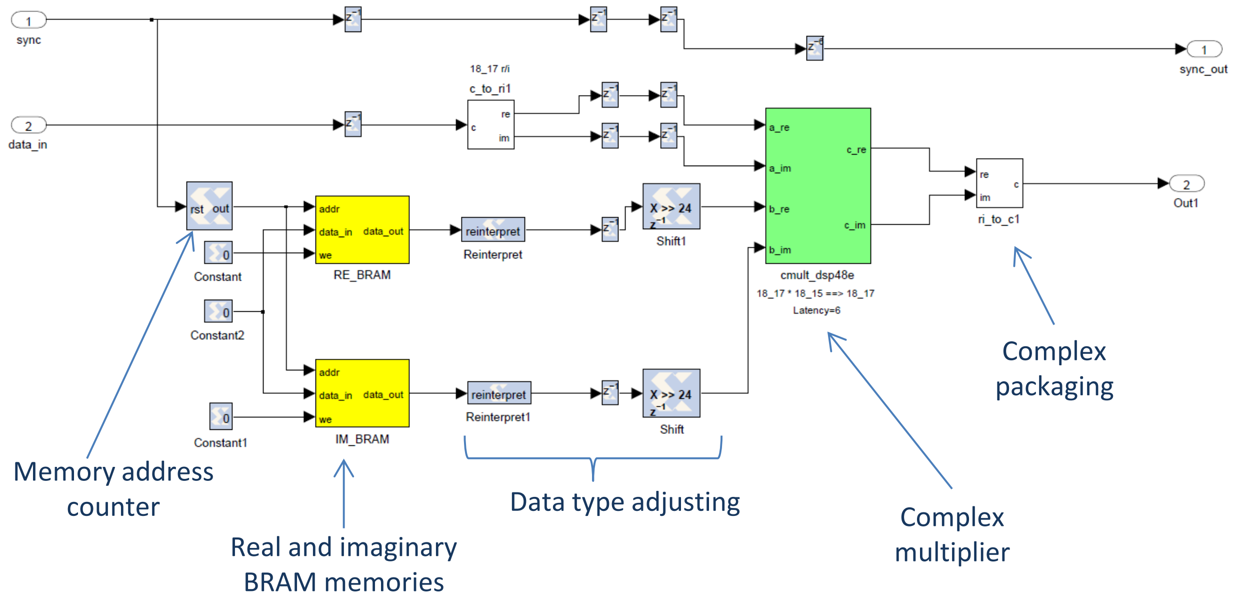


Figure 18: Vector complex multiplier (VCM) block diagram.

The model of Figure 14 could be compiled for an ADC clock speed of up to 500MHz (1GSPS) or a FPGA clock speed of up to 125MHZ so the speed limitation came first from the FPGA maximum clock speed that ensures the correct propagation of the signals within the FPGA fabric. Further optimization on the placing of the FPGA resources may increase the maximum bandwidth.

3.3. Software

A number of python scripts were written to set and read the data from the ROACH. Special programs were designed to perform the calibration which also required the control of the RF and LO frequency synthesizer. The calibration and SRR measurement codes are included in the annex.

4. Implementation

4.1. Test setup

Figure 19 shows the measuring test setup block diagram. Two signal synthesizers were used as LO and RF sources. A noise source was used to provide a white noise floor, which was coupled to the test tone to improve the ADC performance as described by (Morgan and Fisher, 2010), and to emulate a typical radio astronomical application. The analog plate (Figure 10) has two heaters to increase its temperature so the calibration thermal stability can be measured. No particular care was taken to match cable length or to choose matched-pairs of mixers, amplifiers or any other component. Both LO and RF synthesizers, as well as the clock generator for the ADCs and ROACH are locked to the same 10MHz reference, although this is not required for sideband separation.

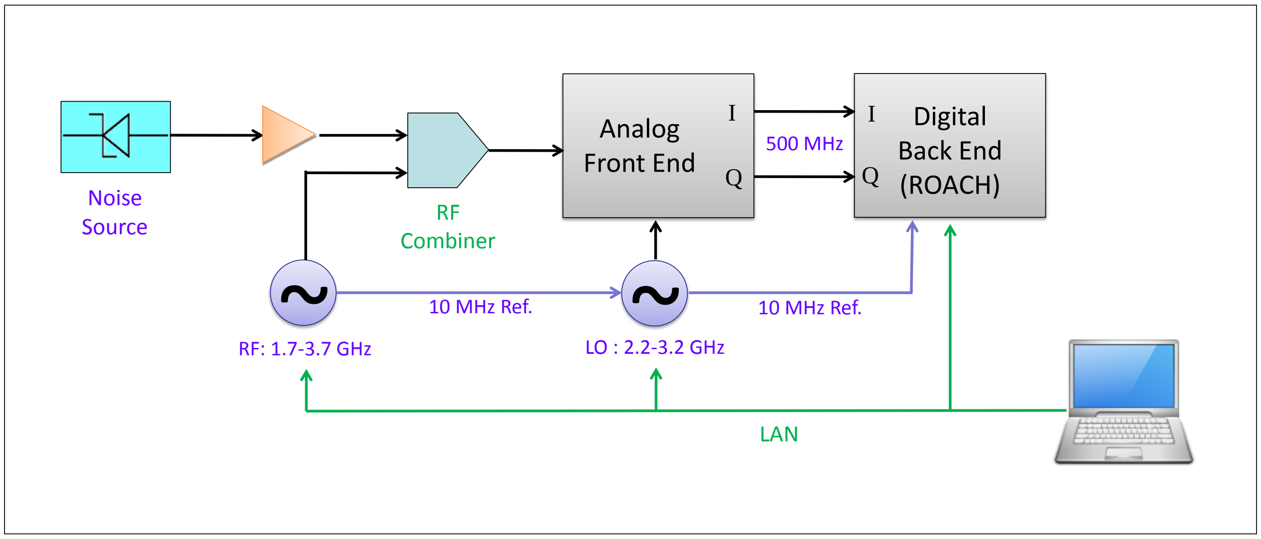


Figure 19: Test setup block diagram. Both LO and RF synthesizers, as well as the clock generator for the ADCs and ROACH are locked to the same 10 MHz reference. A LAN is used to control the instrument and download the spectra.

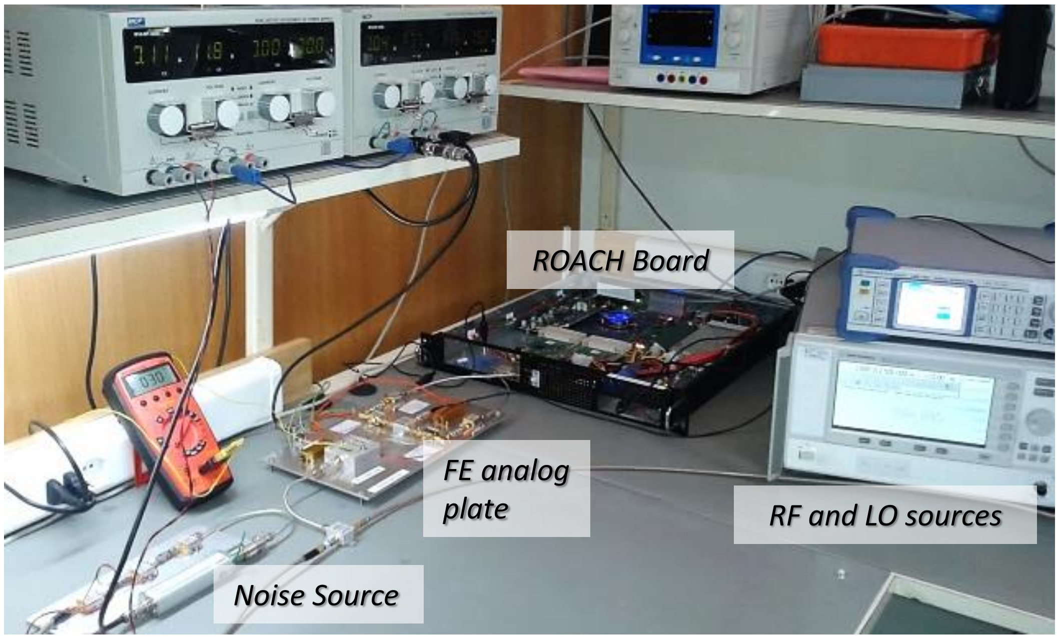


Figure 20: Measuring setup bench. From left to right the noise source and RF combiner can be seen. At the center the front end plate and ROACH assembly are depicted. Computers are not shown.

4.2. Calibration

A second spectrometer was designed to run on the same setup which, instead of accumulating the sideband-separated power spectrum, records the amplitude and phase of 1024 spectral channels (even-numbered channels). Figure 21 shows the model of this calibration spectrometer. The mentioned design directly stores the PFB complex outputs for further analysis on a desktop computer. A test tone is used to measure the relative amplitude and phase of the two analog branches for the 1024 channels. The measurement takes about 40 minutes to be completed. With the collected data the calibration constants C1-C4 are calculated in the following way: C1 and C4 are set to 1+0j while C2 and C3 are determined using the formulas:

and,

$$\frac{C_1}{C_2} = \frac{1}{X} e^{-j(\emptyset_{LSB} - \pi)} \tag{1}$$

$$\frac{C_3}{C_4} = \frac{1}{X} e^{-j(\emptyset_{USB} - \pi)} \tag{2}$$

where X is the amplitude ratio of the two IF branches and $\emptyset_{LSB/USB}$ are the differential phase at the ADC input measured on each sideband (M.A. Morgan & J.R. Fisher, 2010). The calibration coefficients for the odd-numbered channels are calculated by linear interpolation of the even-numbered channels (i.e., the measured data). Finally C1 to C4 are written into the FPGA, a process that takes less than a minute.

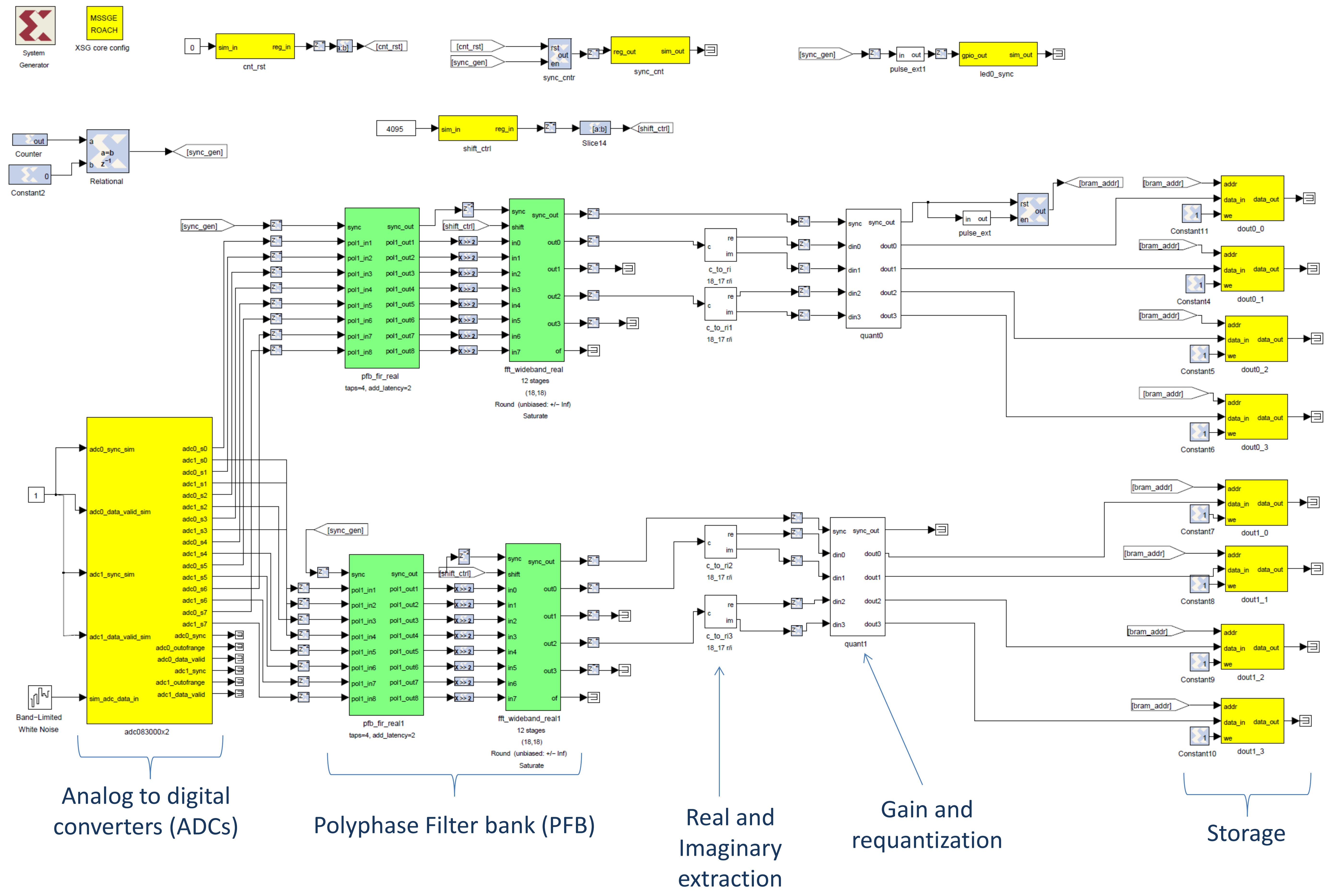


Figure 21: Calibration spectrometer for the measurement of the amplitude and phase imbalance of the analog front end and ADCs.

4.3. Results

4.3.1. Sideband Rejection Measurements

Figure 22 shows the USB and LSB spectra when a full scale (0dBm at the ADC input) test tone is applied on the USB (RF=2.6GHz, LO=2.5GHz). For this example an ideal (uncalibrated) digital hybrid was implemented (C1=C4=1+0j, C2=C3 =0+1j). Spurious signals generated mostly by the ADC are seen around -50dBc limiting the dynamic range of the setup to about 50 dB. The sideband rejection in this example exceeds 20 dB.

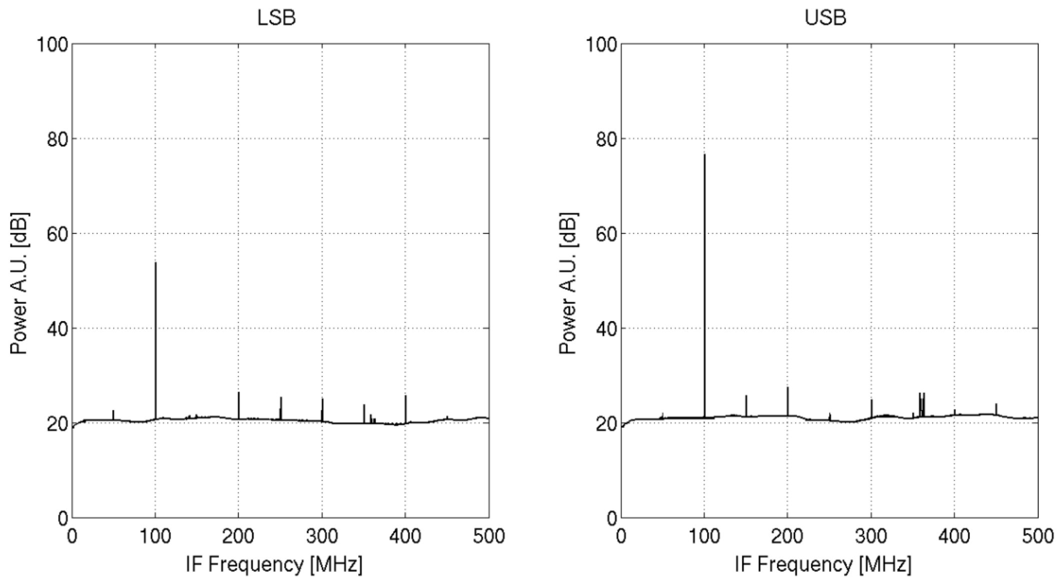


Figure 22: Uncalibrated LSB and USB spectra for a 2.6GHz test tone. LO was set to 2.5GHz. Spurious signals generated in the ADC can be seen at -50dBc on the pass band (USB). The sideband rejection in the example exceeds 20dB.

Figure 23 shows the uncalibrated sideband rejection ratio for a LO of 2.5 GHz and RF frequencies from 2 to 3 GHz. 512 points were measured. Even implementing a perfect IF hybrid the sideband rejection does not exceed 30dB, and averages about 20dB.

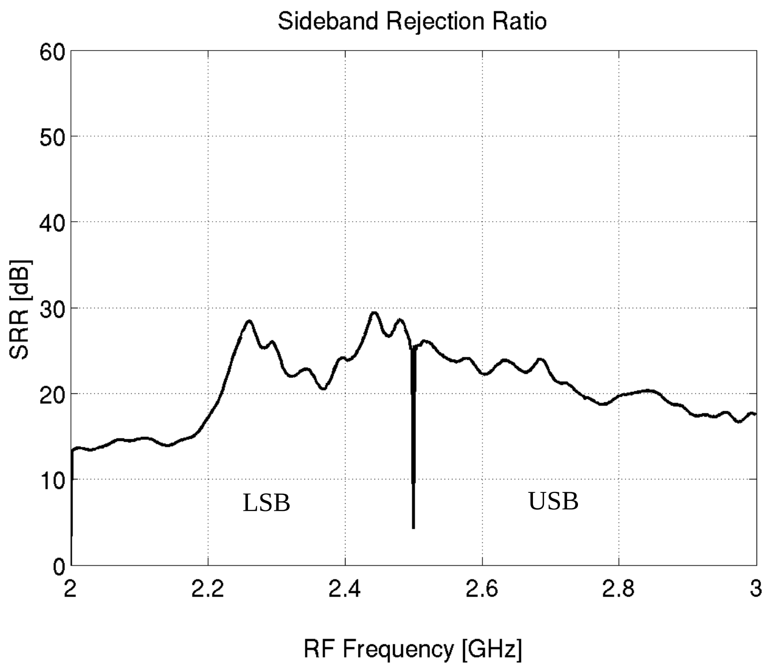


Figure 23: Uncalibrated sideband rejection ratio for a LO of 2.5 GHz and a RF frequency from 2 to 3 GHz.

After calibration, the sideband rejection ratio is measured for every spectral channel by direct calculation of the amplitude ratio of the test tone in both sidebands. This approach can be used since the amplitude imbalance of both signal branches has been equalized, so no additional correction to the direct SRR measurement is needed. Variation of the RF test tone amplitude across the band is also not critical for the accuracy of the SRR measurement although it might degrade the spurious free dynamic range of the spectrometer if the ADCs are driven into saturation or far below its optimal (full scale, 0dBm) amplitude level.

Figure 24 shows the amplitude and phase imbalances of the LSB and USB (LO is set at 2.5GHz). The data shown were taken at 31±1 °C and are used to calculate the constants C2 and C3.

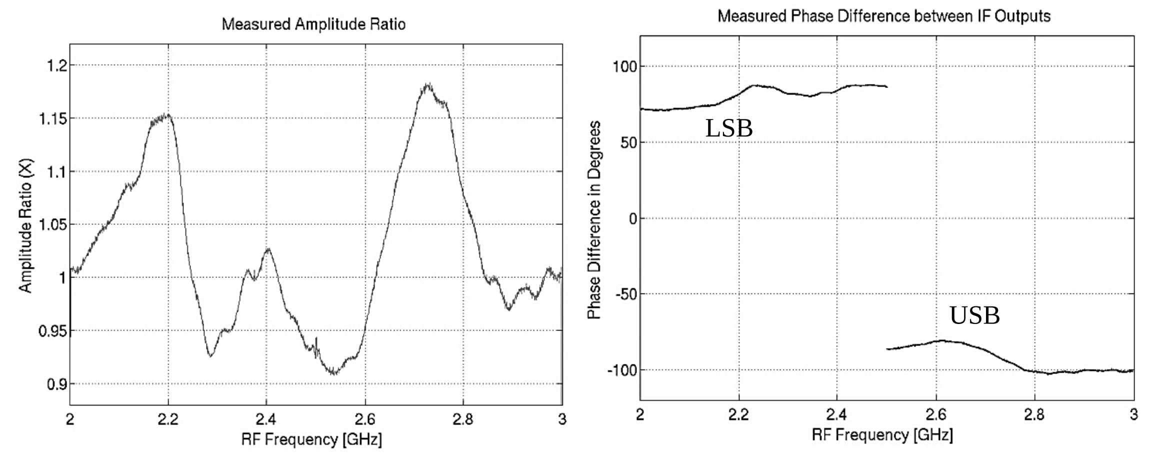


Figure 24: Measured USB and LSB amplitude ratio *X* (left) and phase differences $Ø_{LSB/USB}$ (right). RF input from 2 to 3GHz, LO=2.5GHz

Based on the data shown Figure 24 right, it is possible to calculate the phase unbalance contribution of the LO/RF and IF part independently. The calculation is performed using the expressions (J.R. Fisher & M.A. Morgan, 2008):

$$\emptyset_{LSB} = \emptyset_{IF} + \emptyset_{LO} \tag{3}$$

$$\emptyset_{USB} = \emptyset_{IF} - \emptyset_{LO} \tag{4}$$

where $\emptyset_{LO}$ represents the phase unbalance at the ADC input due to the RF and LO components, and $\emptyset_{IF}$ the signal path mismatch after the mixers. The values are shown in Figure 25. Important unbalances are noticed in both $\emptyset_{LO}$ and $\emptyset_{IF.}$ The result is expected since the IF bandwidth is comparable to the RF maximum bandwidth, so the measurement spanned a substantial portion of the design bandwidth of both RF and IF components.

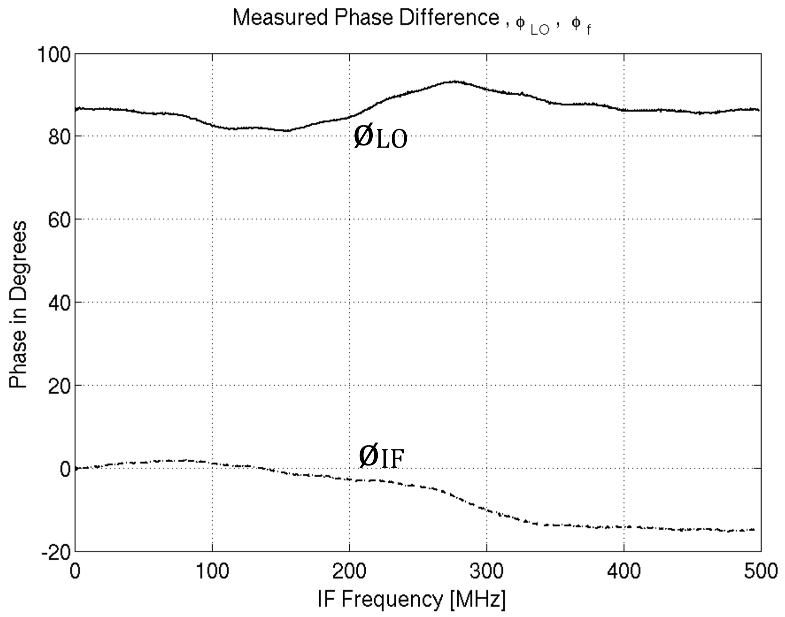


Figure 25: Measured $\emptyset_{LO}$ (solid line) and $\emptyset_{IF}$ (dashed line) contribution to the total phase difference.

Figure 26 shows the sideband rejection ratio for an RF input from 2 to 3GHz, LO=2.5GHz after calibration. The sideband rejection is better than 40dB for the entire band and better than 50dB for most of the band. The drop in SRR close to RF=2.5GHz is due to the AC coupling of the ADCs. The calibration and measurements were performed with the analog front end at 31±1 °C. When the test tone amplitude on the rejected sideband is less than the spurious signal content, the SRR is measured as the ratio of the test tone in the pass band to the stronger spurious in the rejected band, so the measurement is limited by the ADC spurious-free dynamic range to about 50 dB.

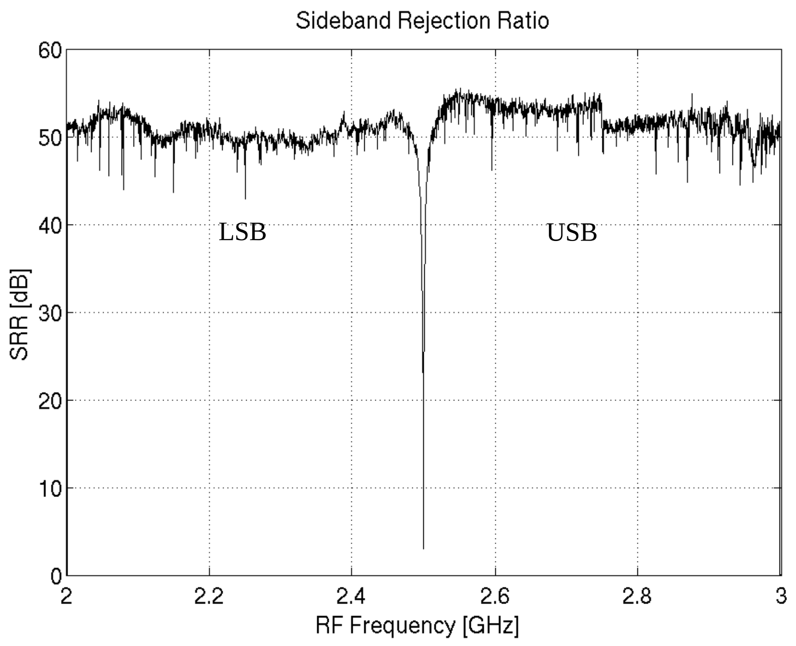


Figure 26: Sideband Rejection Ratio after calibration of all spectral channels. The drop in SRR close to RF=2.5GHz is due to the AC coupling of the ADCs. The lower values closed to 45 dB are associated with stronger spurious signals produced by the ADC.

Figure 27 shows the sideband rejection ratio for a RF band of 1.7 to 3.7 GHz. Two LO frequencies were used at 2.2 and 3.2 GHz. Calibration was performed at each LO. The figure shows that the sideband rejection is similar for different LOs and shows no degradation on the edges of the RF band.

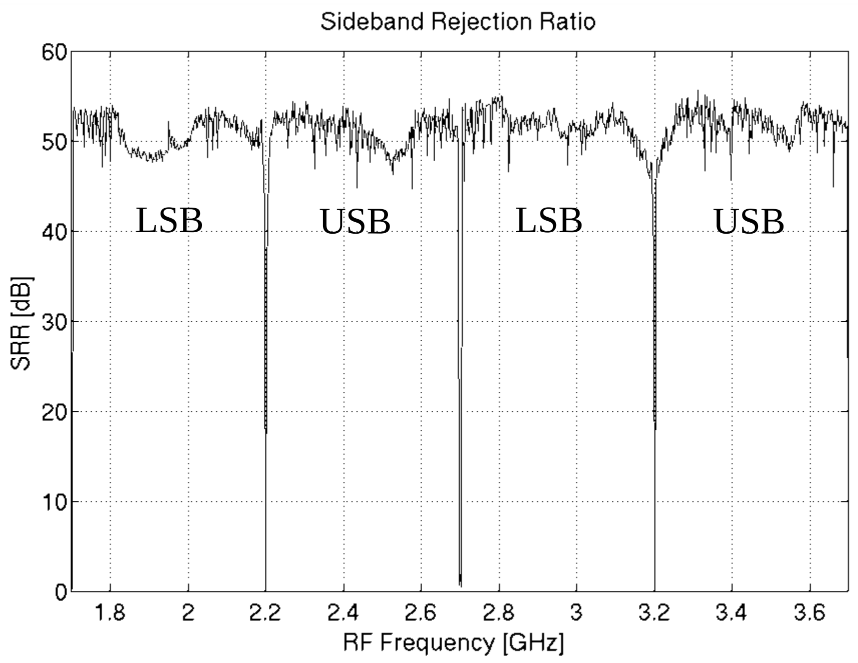
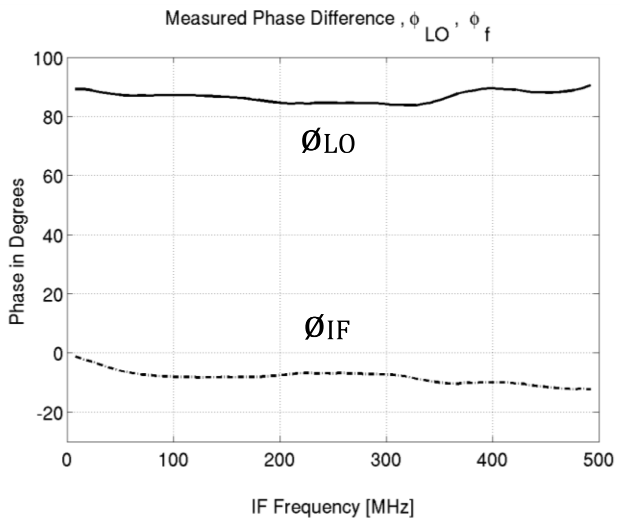
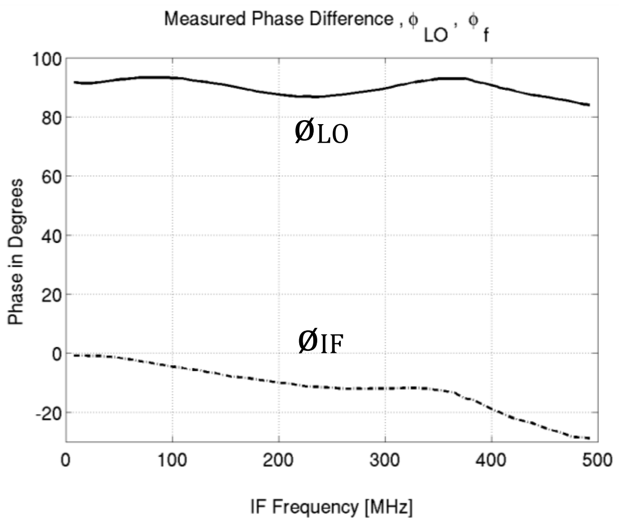


Figure 27: Two LO´s Sideband Rejection Ratio. LO were set to 2.2 and 3.2 GHz while RF ranged from 1.7 to 3.7 GHz. 512 channels per sideband were measured for this figure.

Figure 28 shows the calibration curves for the two LO frequencies of the measurement above. It can be seen that both $Ø_{LO}$ and $Ø_{IF}$ varied when changing LO. It is not clear why $Ø_{IF}$ varies for different LO setting. One possible explanation is due to the variation of the mixers return loss at different LO frequency and power. This could be causing standing waves to appear after the mixers affecting the phase of the IF. More experiments are necessary to better explain these measurements.



(a)



(b)

Figure 28: Two LO´s calibration curves. LO were set to (a) 2.2 and (b) 3.2 GHz. $Ø_{LO}$ and $Ø_{IF}$ varied on each measurement.

4.3.2. Thermal stability measurements

To study the thermal stability of the calibration the analog plate was heated up to 40±1 °C and the SRR was measured for 512 channels. The calibration coefficients remain the same as that measured at 31±1 °C. Figure 29 shows a degradation of up to 20dB for this experiment. The main cause of SRR sensitivity may be the temperature induced phase unbalance due to the low integration of the test front end. Our front end is made of individual components mounted on rather big (31x18cm) plate and connected together. Moreover, two 9cm cables were used between the RF hybrid and the mixers. Better integration, particularly of the RF and LO components, should improve SRR thermal stability.

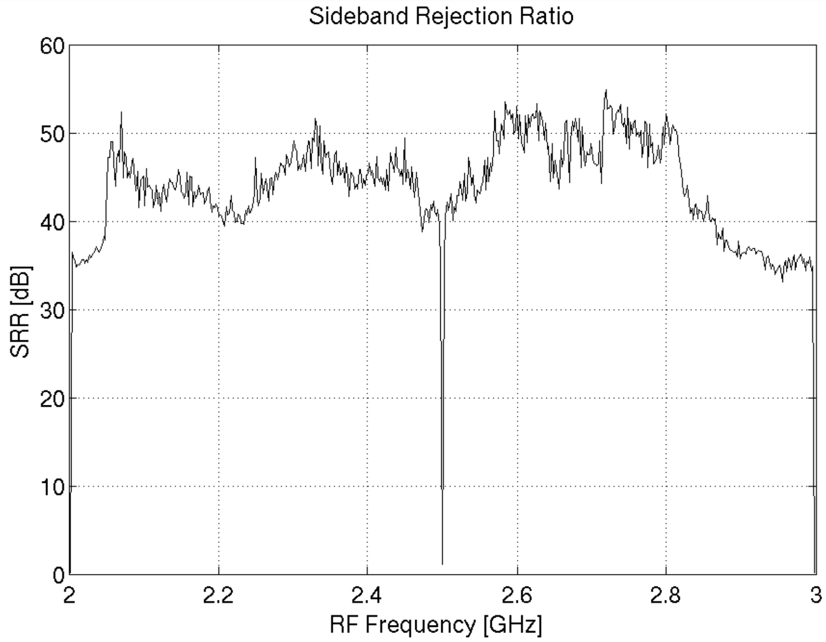


Figure 29: Sideband Rejection Ratio measured at 40±1C with calibration performed at 31±1C. The figure shows the calibration thermal stability of the analog front end used for this experiment. Shorter cables and higher integration particularly of the RF and LO components may improve the thermal stability.

After cooling back to 31±1°C the SRR returns to essentially the same values of Figure 26 suggesting a good long term stability of the calibration. Three such thermal cycles were performed within 48 hours. The SRR was measured for each cycle at 31±1°C. Results are shown in Figure 30.

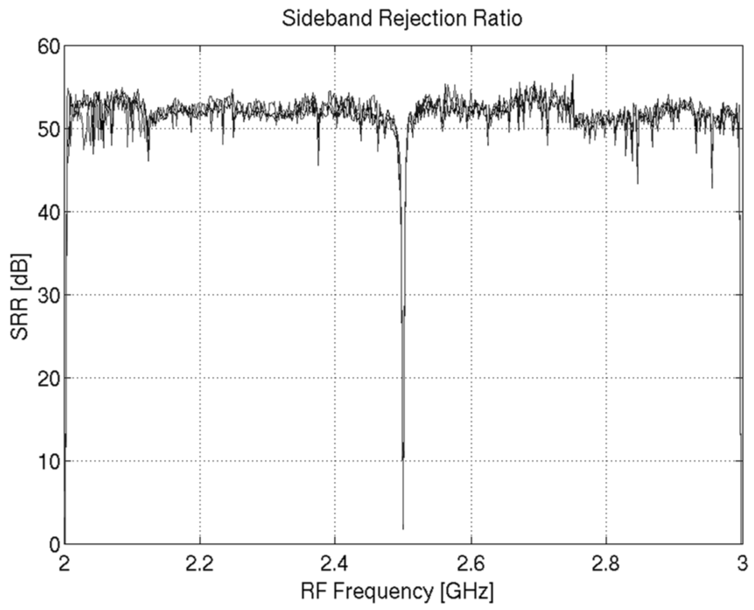


Figure 30: Sideband Rejection Ratio measured at 31±1°C for 3 thermal cycles. The plate was heated up to 40±1°C every time. The calibration was performed only once prior the thermal cycling. The curves overlap almost perfectly showing a good stability of the calibration.

4.3.3. Preliminary results on a millimeter-wave front end

First results have been obtained from the integration of the DSSS with an 86-115 GHz front end. The front end is a custom-made design using an in-house built RF hybrid and LO splitter, two commercial mixers and a Gunn oscillator as LO[9]. The mixers are not matched pairs so significant phase and amplitude unbalances are expected. The RF source for calibration is a 20GHz Agilent PSG series synthesizer followed by a ×6 active multiplier. Figure 31 shows the 3mm front end used for the experiment. The LO injection is at the left where also the IF output filters are seen. At the center the two mixers and the LO splitter are depicted as well as the heating resistor mounted on top of the RF hybrid. The temperature is measured on the LO splitter using a thermocouple (yellow cable).

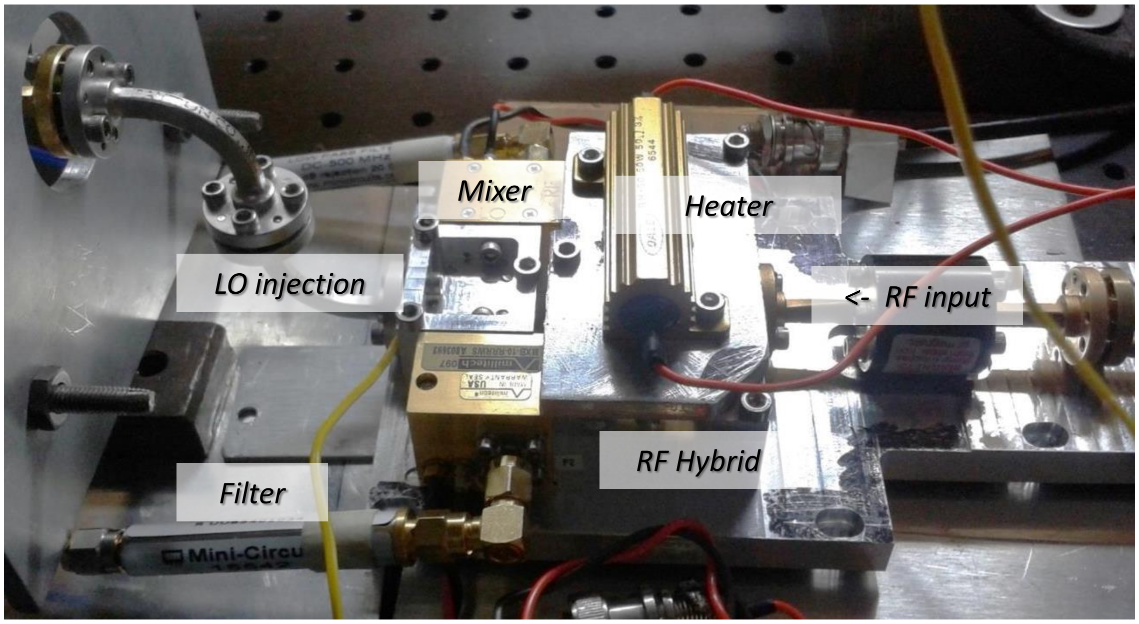


Figure 31: 3mm front end used for the experiment. The LO injection is at the left where also the IF output filters are seen. At the center the two mixers and the LO splitter are depicted as well as the heating resistor mounted on top of the RF hybrid.

A number of difficulties were faced during integration, mainly due the harmonic content of the mixers, high phase noise of the Gunn LO and the presence of RFI on the IF band. On one hand the harmonic content of mixers and the RFI prevented us for measuring sideband rejections above 40-45 dB, since at that level the spurious content was generally above the rejected test tone. On the other hand phase noise of the LO complicated the calibration process due to the way the calibration measurement is implemented. Nevertheless we were able to obtain good SRRs

[9] Design, construction and testing of a 2SB-configuration receiver for the small millimeter-wave telescope (SMWT), PhD Thesis, Rafael Rodrigues (in preparation), soon available on: http://www.das.uchile.cl/lab_mwl/publications.html

between 40 and 45 dB for the entire band. Figure 32 shows the SRR for a RF frequency of 100.5 to 101.5 GHz. The calibration was performed for 64 channels with the rest filled by linear interpolation. Calibration and SRR measurements were done with the front end at a temperature of 26 C. The wider gap at IF=0 is caused in this case by the AC coupling of the mixers, which have a 0.1-3 GHz IF.

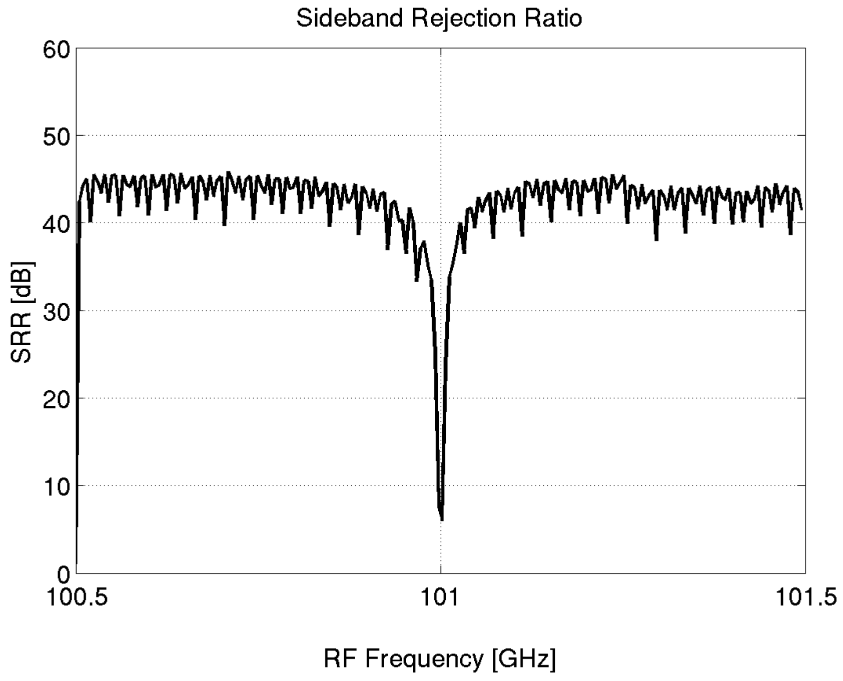


Figure 32: Sideband Rejection Ratio at 201 points after calibration for 64 points. The LO is at 101.0 GHz and the test tone ranged from 100.5 to 101.5 GHz. Calibration and SRR measurement were performed at a temperature of 26C.

Figure 33 shows a typical LSB and USB spectra with the calibration applied. The LO is set to 101.0 GHz while the test tone is at 101.350 GHz. RFI is seen around 100MHz while the spurious signals around 200MHz are mainly due to harmonics and digital artifacts. The test tone is applied on the USB and is rejected more than 40 dB on the LSB. The USB spectrum shows the strong phase noise associated with the LO.

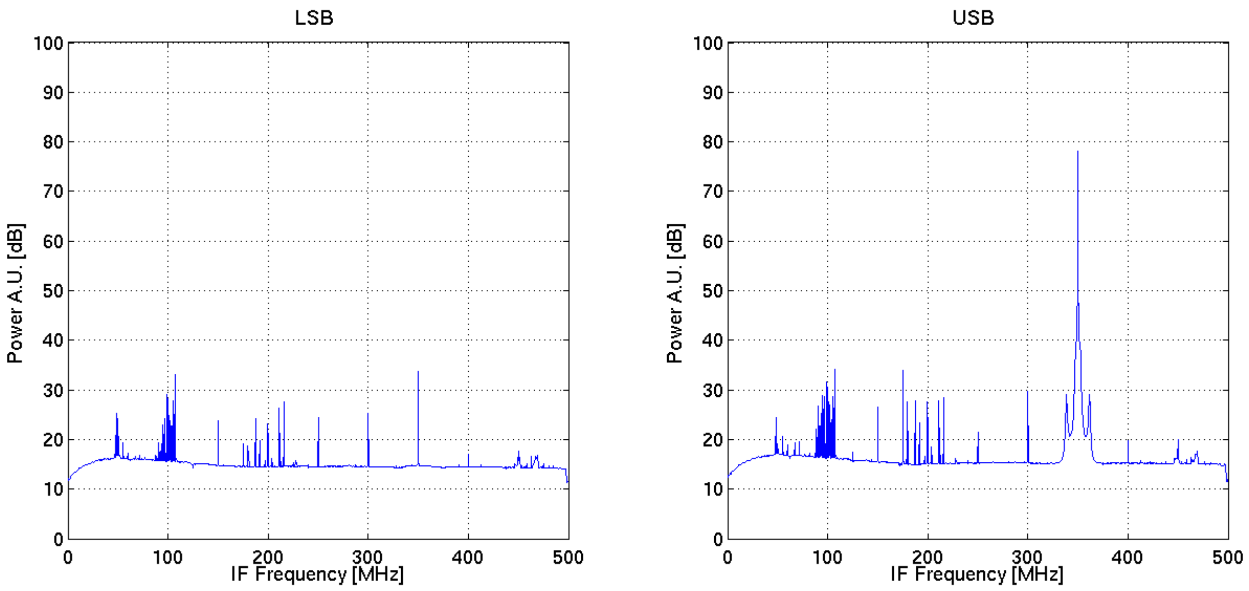


Figure 33: LSB and USB spectra. The LO is set at 101.0 GHz while the test tone to 101.350 GHz. RFI is seen around 100MHZ and the spurious signals around 200MHz are mainly due to harmonics and digital artifacts.

The calibration process is complicated by the phase noise since the calibration spectrometer stores instantaneous spectra at a high speed on the BRAM memories, which are readout at a lower speed from the control computer. Since there is no flow control of the data, it is possible that one read spectrum on the computer has parts coming from different (successive) instantaneous spectra stored by the spectrometer.
If the down-converted test tone phase and amplitude are stable no effect is perceived, since all instantaneous spectra are basically equal. If the test tone has fast amplitude or phase fluctuations, the computer will not acquire accurate data. The result is seen as a noisy, irregular, calibration measurement. Post processing of the calibration data was required to achieve the results of Figure 32. Many calibration points were flagged out and completed through interpolation of the closer well-acquired data. A moving window data selection algorithm was written to perform this task. Work is being done to replace the LO with one of better performance as well as to develop a calibration spectrometer with data flow control and integration capabilities.

The thermal stability was measured by heating the RF hybrid and mixers up to 36C. No strong SRR degradation was noticed in comparison with the results of the lower frequency test front end (Figure 29). The improved stability can be explained due to the fact that the IF band is now small compared with the RF band, while in the former case was comparable. The RF hybrid and mixers are design to work from 85 to 115GHz, so the 1GHz IF represents only a 3.3% of the RF band, while in the former case, the IF represented more than a 45% of the RF bandwidth. Also the mm-wave front end is well integrated and no heating was applied to the IF plate in this case.

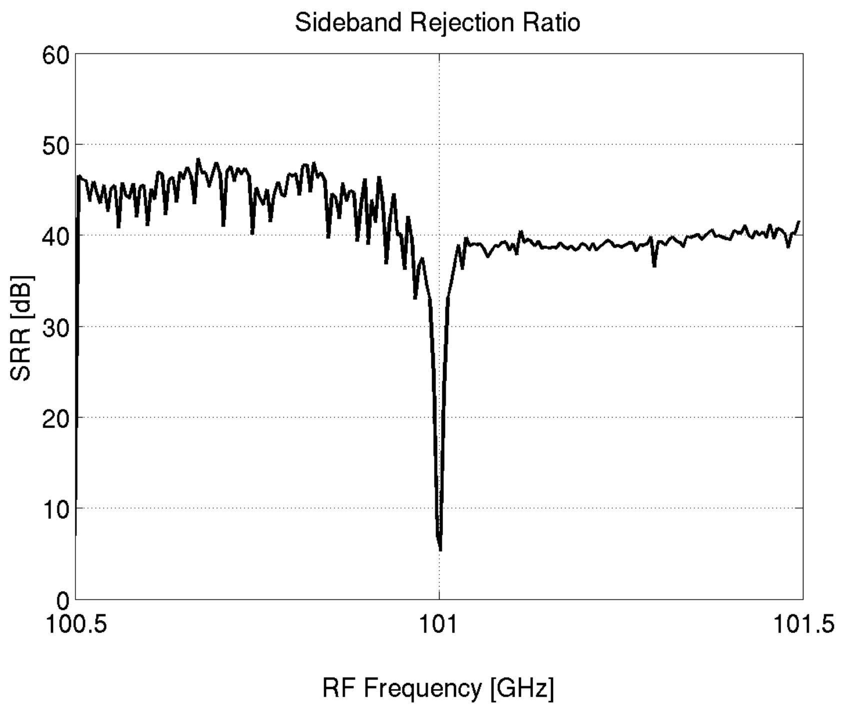


Figure 34: 201 points Sideband Rejection Ratio after 64 points calibration. The LO is at 101.0 GHz and the test tone ranged from 100.5 to 101.5 GHz. Calibration was done at a temperature of 26C while SRR measurement was performed at 36C.

The degradation is only noticed in the USB where the SRR was reduced by about 6 dB. As we discussed above the SRR measurement is limited by the harmonic-free dynamic range of the setup, so the observed degradation has to be interpreted as a lower limit. Probably both sidebands degraded, but only USB degraded enough to be detected by the setup. Nevertheless, it is difficult to explain this disparate degradation, particularly giving the small IF/RF ratio of this receiver. An improved setup of both analog and digital parts, as well as more detailed measurements over wider RF bandwidths are necessary to better understand the performance of this receiver. This will be part of a future work.

5. Future work and astronomical testing

Future work will be devoted to increase the IF bandwidth with a goal of 1.0 to 1.5 GHz per sideband as well as to better understand and improve the performance of the mm-wave digital sideband separating receiver. The integration of the mm-wave receiver into the 1.2m Southern Millimeter Wave Telescope[10] is also planned for near future. This upgrade will allow optimal observation of multiple CO lines. An example science case is the simultaneous observation of ^{12}CO and ^{13}CO in the galactic center. In this interstellar environment the intensity ratio ^{12}CO/^{13}CO can be as low as 5 and the velocity range in excess of +/- 300 Km/s producing line widths of more than 250MHz. Having each line in one sideband, a high sideband rejection will be critical to be able to extract the exact line profiles all the way down to the continuum noise baseline.

[10] The 1.2m Southern Millimeter Wave Telescope is a 85-115GHz radio telescope run by our group at the Cerro Calán National Observatory. http://www.das.uchile.cl/lab_mwl/project.html

6. Conclusion

The digitalization of radio-astronomy instrumentation has been the trend during the last decades. Analog filter banks or acousto-optic spectrometers are no longer built and digital technology is the common solution for new back ends. With the increasing power of digital hardware mainly fast, high resolution ADCs and FPGA technology the digitalization of receiver front ends has become an option. This new approach promises to reduce complexity and increase the performance of the next generation instruments. In this work a concrete step on digitalization of radio-astronomy receivers has been taken building a real-time, calibrated, digital sideband separating spectrometer.

A 4GHz front end was used to test the spectrometer. Two 500MHz IF channels were processed achieving a sideband rejection ratio better than 40 dB over the entire bandwidth and better than 50dB for most of the band. The bandwidth and spectral resolution are competitive for astronomical applications and the sideband rejection is 20 to 30dB better than current millimeter wave sideband separating receivers.

Preliminary tests were done at 101GHz, using a 85-115 GHz front end. The results were similar to those obtained at lower frequencies with an SRR above 40dB for most of the band. This test shows the applicability of the technic at millimeter wavelengths.

The work demonstrates that the use of fast ADCs and FPGA based platforms to perform signal processing currently implemented by analog means can substantially increase the performance of sideband separating receivers.

Appendix A: Python Scripts

Calibration_readout.py

```
# !/usr/bin/env python
# This script was writen by Ricardo Finger et.al. (rfinger@das.uchile.cl).
it is used to get the data to calibrate the sideband separating
spectrometer.
# it works with the ssm_dsp_dif_sin_acc_2013_Jan_04_1441.bof
# You need to have KATCP and CORR installed. Get them from
http://pypi.python.org/pypi/katcp and http://casper.berkeley.edu/
# We used pieces of code writen by Jason Manley (2009) for the CASPER
tutorial 3.

# TO DO: add support for determining ADC input level

import corr,time,struct,sys,logging,Gnuplot,array, telnetlib, valon_synth
from math import *

katcp_port=7147
all_data=open('all_data.dat','w')
cal_data=open('cal_data.dat','w')

def exit_fail():
    print 'FAILURE DETECTED. Log entries:\n',lh.printMessages()
    try:
        fpga.stop()
    except: pass
    raise
    exit()

def exit_clean():
    try:
        fpga.stop()
    except: pass
    exit()

def get_data():
# Channel I
    re_0i=struct.unpack('>512q',fpga.read('dout0_0',512*8,0))
    im_0i=struct.unpack('>512q',fpga.read('dout0_1',512*8,0))
    re_2i=struct.unpack('>512q',fpga.read('dout0_2',512*8,0))
    im_2i=struct.unpack('>512q',fpga.read('dout0_3',512*8,0))
# Channel Q
    re_0q=struct.unpack('>512q',fpga.read('dout1_0',512*8,0))
    im_0q=struct.unpack('>512q',fpga.read('dout1_1',512*8,0))
    re_2q=struct.unpack('>512q',fpga.read('dout1_2',512*8,0))
    im_2q=struct.unpack('>512q',fpga.read('dout1_3',512*8,0))

    spec_i=[]
    spec_q=[]
    power_spec_i=[]
    power_spec_q=[]
```

```
#   The "+1" in the "for" is to avoid division by zero.
#   The induced error should be neglectable.
#   The "+1" can be removed is you have a detectable noise floor.
    for i in range(512):
        spec_i.append(float(re_0i[i]+1)/(2**18))
        spec_i.append(float(im_0i[i]+1)/(2**18))
        spec_i.append(float(re_2i[i]+1)/(2**18))
        spec_i.append(float(im_2i[i]+1)/(2**18))
        spec_q.append(float(re_0q[i]+1)/(2**18))
        spec_q.append(float(im_0q[i]+1)/(2**18))
        spec_q.append(float(re_2q[i]+1)/(2**18))
        spec_q.append(float(im_2q[i]+1)/(2**18))

    for i in range(0,4*512,2):
        power_spec_i.append(10*log10(1+(spec_i[i]**2+spec_i[i+1]**2)**1))
        power_spec_q.append(10*log10(1+(spec_q[i]**2+spec_q[i+1]**2)**1))

    return spec_i, spec_q, power_spec_i, power_spec_q

def c_angle(re,im):#complex angle / evaluates atan(Im/Re) for the 4
cuadrants
# initializing
    out=0
    if re==0:
      re=10**-20
    if im==0:
      im=10**-20
# Angle calculation
    if im>=0.0 and re>=0.0:
        out=atan(im/re)
    if im>=0.0 and re<=0.0:
        out=pi/2+atan(abs(re)/im)
    if im<=0.0 and re<=0.0:
        out=pi+atan(abs(im)/abs(re))
    if im<=0.0 and re>=0.0:
        out=(3*pi/2)+atan(re/abs(im))
    return out # the output is in radians

def trunca(f, n):
    '''Truncates/pads a float f to n decimal places without rounding'''
    slen = len('%.*f' % (n, f))
    return str(f)[:slen]

################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser
    p = OptionParser()
    p.set_usage('calibration_readout.py <ROACH_HOSTNAME_or_IP> [options]')
    p.set_description(__doc__)
    p.add_option('-s', '--skip', dest='skip', action='store_true',
        help='Skip reprogramming the FPGA and configuring EQ.')
    p.add_option('-b', '--bof', dest='boffile',type='str', default='',
        help='Specify the bof file to load')
    opts, args = p.parse_args(sys.argv[1:])
    bitstream = opts.boffile
    if args==[]:
        print 'Please specify a ROACH board. Run with the -h flag to see
all options.\nExiting.'
        exit()
    else:
        roach = args[0]
try:
    loggers = []
    lh=corr.log_handlers.DebugLogHandler()
    logger = logging.getLogger(roach)
    logger.addHandler(lh)
    logger.setLevel(10)

    print('Connecting to server %s on port %i... '%(roach,katcp_port)),
    fpga = corr.katcp_wrapper.FpgaClient(roach, katcp_port,
timeout=10,logger=logger)
    time.sleep(1)

    if fpga.is_connected():
        print 'ok\n'
    else:
        print 'ERROR connecting to server %s on port
%i.\n'%(roach,katcp_port)
        exit_fail()

    print '------------------------'
    print 'Programming FPGA with %s...' %bitstream,
    if not opts.skip:
        fpga.progdev(bitstream)
        print 'done'
    else:
        print 'Skipped.'

    time.sleep(1)

    print 'Configuring FFT shift register...',
    fpga.write('shift_ctrl','\x00\x00\x0f\xff')
    print 'done'
#
    print 'Resetting counters...',
    fpga.write_int('cnt_rst',1)
    fpga.write_int('cnt_rst',0)
    print 'done'

##############  Start of the measurement  ##################

    ti=time.time()
    bw=float(trunc(fpga.est_brd_clk())*4)
    LO=input('LO frequency GHz? ')*1000#LO is in MHz
    ch=input('number of channels(2^?) ')
#    rys = telnetlib.Telnet("172.17.89.49",5025)
#    valon=valon_synth.Synthesizer('/dev/ttyUSB0')
    agi = telnetlib.Telnet("172.17.89.50",5023)
#    agi.write("freq 2.0ghz\r\n")
#    agi.write("power -12dbm\r\n")
#    agi.write("output on\r\n")
#    rys.write("freq "+str(float(LO)/1000)+"ghz\r\n")
#    rys.write("output on\r\n")

    all_data.write('#Canal'+' '+'real i'.rjust(7)+' '+'imag i'.rjust(7)+'
'+'real q'.rjust(7)+' '+'imag q'.rjust(7)+' '+'amp_i'.rjust(7)+'
'+'amp_q'.rjust(7)+' '+'phase_i'.rjust(7)+' '+'phase_q'.rjust(7)+'
'+'amp_rat'.rjust(7)+' '+'ang_dif'.rjust(7)+' '+'phi'.rjust(7)+' \n')

#   set up the figures to be plotted
    g0 = Gnuplot.Gnuplot(debug=0)
    g1 = Gnuplot.Gnuplot(debug=0)

    g0.clear()
    g0.title('i Spectrum '+bitstream+' | Max frequency = '+str(bw)+' MHz')
    g0.xlabel('Channel #')
    g0.ylabel('power AU (dB)')
    g0('set style data linespoints')
    g0('set yrange [0:100]')
    g0('set xrange [-50:1074]')
    g0('set xtics 256')
    g0('set grid y')
    g0('set grid x')

    g1.clear()
    g1.title('q Spectrum'+bitstream+' | Max frequency = '+str(bw)+' MHz')
    g1.xlabel('Channel #')
    g1.ylabel('power AU (dB)')
    g1('set style data linespoints')
    g1('set yrange [0:100]')
    g1('set xrange [-50:1074]')
    g1('set xtics 256')
    g1('set grid y')
    g1('set grid x')
#   End setting figures to be plotted

#############   Begin measureing USB  ##############
    wait=0.5
    data_usb=[]
    print('Measuring USB')
    print('freq   amp_i   amp_q  angle_i angle_q  amp_i/q phi   phase_dif
')
    for i in range(1*1024/(2**ch),1023,1024/(2**ch)):#USB
                t=time.time()
                if i/20==float(i)/20:
               print(str(trunca((t-ti)/60,2))+' minutes'+'
'+str(trunca(float(i)*100.0/2044,2))+' %')
           #rys.write("freq "+ str(LO+(2*i)*bw/2048) +"mhz\r\n")
             agi.write("freq "+ str(LO+(2*i)*bw/2048) +"mhz\r\n")
           time.sleep(wait)
                spec_i, spec_q, power_spec_i, power_spec_q = get_data()
           amp_i=((spec_i[2*i])**2+(spec_i[2*i+1])**2)**0.5
           amp_q=((spec_q[2*i])**2+(spec_q[2*i+1])**2)**0.5
           angle_i=c_angle(spec_i[2*i],spec_i[2*i+1])*180/(pi) #in degrees
           angle_q=c_angle(spec_q[2*i],spec_q[2*i+1])*180/(pi) #in degrees

      phi=c_angle(spec_i[2*i]*spec_q[2*i]+spec_i[2*i+1]*spec_q[2*i+1],spec
_q[2*i]*spec_i[2*i+1]-spec_i[2*i]*spec_q[2*i+1])*180/(pi) #phi_f+/-phi_LO
or Phi_USB from eq_13(2008)
           amp_ratio=amp_i/amp_q# X from eq_9(2008)
           phase_dif=(angle_i-angle_q)# This should be equal to phi

                if phase_dif<0:
                   phase_dif=360+phase_dif
                data_usb.append([1/(amp_ratio),LO+(2*i)*bw/2048])
           data_usb.append([180-phase_dif,LO+(2*i)*bw/2048])

########### Only for printing on screen and on "all_data" ########
                amp_i=trunca(amp_i,2)
                amp_q=trunca(amp_q,2)
                angle_i=trunca(angle_i,2)
                angle_q=trunca(angle_q,2)
           amp_ratio=trunca(amp_ratio,2)
           phase_dif=trunca(phase_dif,2)
           phi=trunca(phi,2)
           print(str(trunca(LO+(2*i)*bw/2048,0))+' '+str(amp_i).rjust(7)+'
'+str(amp_q).rjust(7)+' '+str(angle_i).rjust(7)+'
'+str(angle_q).rjust(7)+' '+str(amp_ratio).rjust(7)+'
'+str(phi).rjust(7)+' '+str(phase_dif).rjust(7))
                all_data.write(repr(2*i).rjust(6)+'
'+trunca(spec_i[2*i],2).rjust(7)+' '+trunca(spec_i[2*i+1],2).rjust(7)+'
'+trunca(spec_q[2*i],2).rjust(7)+' '+trunca(spec_q[2*i+1],2).rjust(7)+'
'+(amp_i).rjust(7)+' '+(amp_q).rjust(7)+' '+(angle_i).rjust(7)+'
'+(angle_q).rjust(7)+' '+(amp_ratio).rjust(7)+' '+(phase_dif).rjust(7)+'
'+phi.rjust(7)+' \n')
########### plotting ######################
                g0.plot(power_spec_i)
                g1.plot(power_spec_q)

###############   Begin measurement LSB  ################
    all_data.write('lower_sideband'+' \n')
    all_data.write('#Canal'+' '+'real i'.rjust(7)+' '+'imag i'.rjust(7)+'
'+'real q'.rjust(7)+' '+'imag q'.rjust(7)+' '+'amp_i'.rjust(7)+'
'+'amp_q'.rjust(7)+' '+'phase_i'.rjust(7)+' '+'phase_q'.rjust(7)+'
'+'amp_rat'.rjust(7)+' '+'ang_dif'.rjust(7)+' '+'phi'.rjust(7)+' \n')
    data_lsb=[]
    print('Measuring LSB')
    print('freq   amp_i   amp_q  angle_i angle_q  amp_i/q phi   phase_dif
')
    for i in range(1*1024/(2**ch),1023,1024/(2**ch)):#LSB
           t=time.time()
                if i/20==float(i)/20:
               print(str(trunca((t-ti)/60,2))+' minutes'+'
'+str(trunca(50+float(i)*100.0/2044,2))+' %')
           #rys.write("freq "+ str(LO-(2*i)*bw/2048) +"mhz\r\n")
             agi.write("freq "+ str(LO-(2*i)*bw/2048) +"mhz\r\n")
           time.sleep(wait)
                spec_i, spec_q, power_spec_i, power_spec_q = get_data()
           amp_i=((spec_i[2*i])**2+(spec_i[2*i+1])**2)**0.5
           amp_q=((spec_q[2*i])**2+(spec_q[2*i+1])**2)**0.5
           angle_i=c_angle(spec_i[2*i],spec_i[2*i+1])*180/(pi) #en grados
           angle_q=c_angle(spec_q[2*i],spec_q[2*i+1])*180/(pi) #en grados

      phi=c_angle(spec_i[2*i]*spec_q[2*i]+spec_i[2*i+1]*spec_q[2*i+1],spec
_q[2*i]*spec_i[2*i+1]-spec_i[2*i]*spec_q[2*i+1])*180/(pi)#phi_f+/-phi_LO
or Phi_USB from eq_13(2008)
           amp_ratio=amp_i/amp_q# X from eq_9(2008)
           phase_dif=(angle_i-angle_q)# This should be equal to phi

                if phase_dif<0:
                    phase_dif=360+phase_dif
                data_lsb.append([amp_ratio,LO-(2*i)*bw/2048])
           data_lsb.append([phase_dif-180,LO-(2*i)*bw/2048])

########### Only for printing on screen and on "all_data" ########
                amp_i=trunca(amp_i,2)
                amp_q=trunca(amp_q,2)
                angle_i=trunca(angle_i,2)
                angle_q=trunca(angle_q,2)
           amp_ratio=trunca(amp_ratio,2)
           phase_dif=trunca(phase_dif,2)
           phi=trunca(phi,2)
           print(str(trunca(LO-(2*i)*bw/2048,0))+' '+str(amp_i).rjust(7)+'
'+str(amp_q).rjust(7)+' '+str(angle_i).rjust(7)+'
'+str(angle_q).rjust(7)+' '+str(amp_ratio).rjust(7)+'
'+str(phi).rjust(7)+' '+str(phase_dif).rjust(7))
                all_data.write(repr(2*i).rjust(6)+'
'+trunca(spec_i[2*i],2).rjust(7)+' '+trunca(spec_i[2*i+1],2).rjust(7)+'
'+trunca(spec_q[2*i],2).rjust(7)+' '+trunca(spec_q[2*i+1],2).rjust(7)+'
'+(amp_i).rjust(7)+' '+(amp_q).rjust(7)+' '+(angle_i).rjust(7)+'
'+(angle_q).rjust(7)+' '+(amp_ratio).rjust(7)+' '+(phase_dif).rjust(7)+'
'+phi.rjust(7)+' \n')
########### plotting ######################
                g0.plot(power_spec_i)
                g1.plot(power_spec_q)
########### Writing calibration data ########
#    cal_data.write('#lower_sideband'+' \n')
    for i in range(0,len(data_lsb),2):
        cal_data.write('0 '+str(data_lsb[i][0])+' '+str(data_lsb[i][1])+'
\n')# stores: amp_ratio     IF_freq
    for i in range(1,len(data_lsb),2):
      cal_data.write('0 '+str(data_lsb[i][0])+' '+str(data_lsb[i][1])+'
\n')# stores: phase_dif-180  IF_freq
#    cal_data.write('#upper_sideband'+' \n')
    for i in range(0,len(data_usb),2):
        cal_data.write('0 '+str(data_usb[i][0])+' '+str(data_usb[i][1])+'
\n')# stores: 1/amp_ratio    IF_freq
    for i in range(1,len(data_usb),2):
      cal_data.write('0 '+str(data_usb[i][0])+' '+str(data_usb[i][1])+'
\n')# stores: 180-phase_dif  IF_freq
    tf=time.time()
    print('Done!      Total time='+repr(trunca((tf-ti)/60,1))+' minutes')

except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()

exit_clean()
```

SRR_measurement.py

```
# !/usr/bin/env python
# This script reads the calibration data "cal_data.dat" and calibrates the
sideband separating spectrometer
# You need to have KATCP and CORR installed. Get them from
http://pypi.python.org/pypi/katcp and http://casper.berkeley.edu/
# We use pieces of code writen by Jason Manley (2009) for the CASPER
tutorial 3.

# TO DO: add support for determining ADC input level

import corr, time, struct, sys, logging, Gnuplot, valon_synth, telnetlib,
numpy as np
from math import *

bitstream = 'No_bof_file_error'
katcp_port=7147

t1 = time.time()  #Starting time.

print ('-------------------------------------------------------')
print ('        Sideband Rejection Ration Measurement           ')
print ('-------------------------------------------------------')

def hhmmss(seconds):
    hh = seconds // 3600
    mm = (seconds % 3600)//60
    ss = (seconds %3600) %60
    return hh,mm,ss

def exit_fail():
    print 'FAILURE DETECTED. Log entries:\n',lh.printMessages()
    try:
        fpga.stop()
    except: pass
    raise
    exit()

def exit_clean():
    try:
        fpga.stop()
    except: pass
    exit()

def get_data():
    #get the data...
    acc_n = fpga.read_uint('acc_cnt')
# Channel I
    a_0l=struct.unpack('>512Q',fpga.read('dout0_0',512*8,0))
    a_1l=struct.unpack('>512Q',fpga.read('dout0_1',512*8,0))
    a_2l=struct.unpack('>512Q',fpga.read('dout0_2',512*8,0))
    a_3l=struct.unpack('>512Q',fpga.read('dout0_3',512*8,0))
# Channel Q
    a_0m=struct.unpack('>512Q',fpga.read('dout1_0',512*8,0))
    a_1m=struct.unpack('>512Q',fpga.read('dout1_1',512*8,0))
    a_2m=struct.unpack('>512Q',fpga.read('dout1_2',512*8,0))
    a_3m=struct.unpack('>512Q',fpga.read('dout1_3',512*8,0))

    interleave_i=[]
    interleave_q=[]
    interleave_log_i=[]
    interleave_log_q=[]

#   The "+1" in the "for" is to avoid division by zero.
#   The induced error should be neglectable.
#   The "+1" can be removed is you have a detectable noise floor.
    for i in range(512):
        interleave_i.append(float(((float(a_0l[i])+1)/(2**24))))
        interleave_i.append(float(((float(a_1l[i])+1)/(2**24))))
        interleave_i.append(float(((float(a_2l[i])+1)/(2**24))))
        interleave_i.append(float(((float(a_3l[i])+1)/(2**24))))
        interleave_q.append(float(((float(a_0m[i])+1)/(2**24))))
        interleave_q.append(float(((float(a_1m[i])+1)/(2**24))))
        interleave_q.append(float(((float(a_2m[i])+1)/(2**24))))
        interleave_q.append(float(((float(a_3m[i])+1)/(2**24))))

    for k in range(4*512):
        interleave_log_i.append(10*log10(interleave_i[k]))
        interleave_log_q.append(10*log10(interleave_q[k]))

    return acc_n, interleave_log_i, interleave_log_q

#############   Opening archives    ###################
srr_datos=open('srr_data.dat','w')
lsb_usb_testtone=open('lsb_usb_testtone.dat','w')

################   START OF MAIN   ################

if __name__ == '__main__':
    from optparse import OptionParser

    p = OptionParser()
    p.set_usage('SRR_measurement.py <ROACH_HOSTNAME_or_IP> [options]')
    p.set_description(__doc__)
    p.add_option('-l', '--acc_len', dest='acc_len',
type='int',default=2*(2**28)/2048,
        help='Set the number of vectors to accumulate between dumps.
default is 2*(2^28)/2048, or just under 2 seconds.')
    p.add_option('-g', '--gain', dest='gain',
type='int',default=0x00001000,
        help='Set the digital gain (6bit quantisation scalar). Default is
0xffffffff (max), good for wideband noise. Set lower for CW tones.')
    p.add_option('-s', '--skip', dest='skip', action='store_true',
        help='Skip reprogramming the FPGA and configuring EQ.')
    p.add_option('-b', '--bof', dest='boffile',type='str', default='',
        help='Specify the bof file to load')
    opts, args = p.parse_args(sys.argv[1:])

    if args==[]:
        print 'Please specify a ROACH board. Run with the -h flag to see
all options.\nExiting.'
        exit()
    else:
        roach = args[0]
    if opts.boffile != '':
        bitstream = opts.boffile

try:
    loggers = []
    lh=corr.log_handlers.DebugLogHandler()
    logger = logging.getLogger(roach)
    logger.addHandler(lh)
    logger.setLevel(10)

    print('Connecting to server %s on port %i... '%(roach,katcp_port)),
    fpga = corr.katcp_wrapper.FpgaClient(roach, katcp_port,
timeout=10,logger=logger)
    time.sleep(1)

    if fpga.is_connected():
        print 'ok\n'
    else:
        print 'ERROR connecting to server %s on port
%i.\n'%(roach,katcp_port)
        exit_fail()

    print '------------------------'
    print 'Programming FPGA with %s...' %bitstream,
    if not opts.skip:
        fpga.progdev(bitstream)
        print 'done'
    else:
        print 'Skipped.'
#
    print 'Configuring FFT shift register...',
    fpga.write('shift_ctrl','\x00\x00\x0f\xff')
    print 'done'
#
    print 'Configuring accumulation period...',
    fpga.write_int('acc_len',opts.acc_len)
    print 'done'
#
    print 'Resetting counters...',
    fpga.write_int('cnt_rst',1)
    fpga.write_int('cnt_rst',0)
    print 'done'
#
    print 'Setting digital gain of all channels to %i...'%opts.gain,
    if not opts.skip:
        fpga.write_int('gain',opts.gain) #write the same gain for all
inputs, all channels
        print 'done'
    else:
        print 'Skipped.'

    bw=float(trunc(fpga.est_brd_clk())*4)
    print ('Detected bandwidth: '+str(fpga.est_brd_clk()*4)+' MHz'+',
Using: '+str(bw)+' MHz')
    LO=input('LO frequency [GHz] ? : ') #GHz
    start=LO-bw/1000
    stop=LO+bw/1000
    print ('Measuring span: '+str(start)+' to '+str(stop)+' GHz')
    points=input('Number of SRR points? :')
    cal_mode=input('Calibration mode? (1:Calibrated, 2:Ideal IF hybrid):
')

########### setting up figures #############

    g0 = Gnuplot.Gnuplot(debug=0)
    g1 = Gnuplot.Gnuplot(debug=0)
    g2 = Gnuplot.Gnuplot(debug=0)

    g0.clear()
    g0.title('Channel I(USB) spectrum using '+bitstream+' | Max frequency
= '+str(bw)+' MHz')
    g0.xlabel('Channel #')
    g0.ylabel('Power AU (dB)')
    g0('set style data linespoints')
    g0('set yrange [0:120]')
    g0('set xrange [-50:2098]')
    g0('set ytics 5')
    g0('set xtics 256')
    g0('set grid y')
    g0('set grid x')

    g1.clear()
    g1.title('Channel Q(LSB)  spectrum using '+bitstream+' | Max frequency
= '+str(bw)+' MHz')
    g1.xlabel('Channel #')
    g1.ylabel('Power AU (dB)')
    g1('set style data linespoints')
    g1('set yrange [0:120]')
    g1('set xrange [-50:2098]')
    g1('set ytics 5')
    g1('set xtics 256')
    g1('set grid y')
    g1('set grid x')

    g2.clear()
    g2.title('Sideband Rejection Ratio using '+bitstream+' | running at
'+str(bw)+' MHz')
    g2.xlabel('RF freq (GHz) - shown LSB and USB -')
    g2.ylabel('SSR (dB)')
    g2('set style data linespoints')
    g2('set yrange [0:60]')
    g2('set xrange ['+str(start-0.05)+":"+str(stop+0.05)+']')
    g2('set ytics 5')
    g2('set grid y')
    g2('set grid x')

##### starting the measurement proccess

    rys = telnetlib.Telnet("172.17.89.49",5025)
#    agi = telnetlib.Telnet("172.17.89.50",5023)
#    valon=valon_synth.Synthesizer('/dev/ttyUSB0')
```

```
    srr=[]
    rf=[]
    tono_usb=[]
    tono_lsb=[]
    canales=[]
    ch=[]
    for i in range(points):
        rf.append(start+((stop-start)*i/points))#in GHz

#####################################################
############  Aquaring calibration data #############
#####################################################
    print 'Aquaring calibration data'
    c2amp=[]
    c2phase=[]
    c3amp=[]
    c3phase=[]
    cal_data=np.loadtxt('cal_data.dat')#calibration data
#ch1 + c2*ch2 = USB
    for i in range (0,(len(cal_data)/4)):
        c2amp.append(cal_data[i,1])# X_lsb
    for i in range ((1*len(cal_data)/4),(2*(len(cal_data)/4))):
        c2phase.append(cal_data[i,1])# phi_lsb - 180
#ch2 + c3*ch1 = LSB
    for i in range ((2*(len(cal_data)/4)),(3*(len(cal_data)/4))):
        c3amp.append(cal_data[i,1])# 1/X_usb
    for i in range ((3*(len(cal_data)/4)),(4*(len(cal_data)/4))):
        c3phase.append(cal_data[i,1])# 180 - phi_usb
    print 'Done'
###############################################################
############  writing calibration data   ######################
###############################################################
    print 'writing calibration data'
    if cal_mode == 1:
        for j in range(0,510):
            print ('Writing coeficient '+str(j)+' of 510'+' please 
wait...')
########################## writing C2 ######################

fpga.write('VCM0_RE_BRAM',struct.pack('>1l',c2amp[2*j]*cos((pi/180)*c2phas
e[2*j])*2**24),4*j)  #ch 0,4,8 ...  (4i)

fpga.write('VCM0_IM_BRAM',struct.pack('>1l',c2amp[2*j]*sin((pi/180)*c2phas
e[2*j])*2**24),4*j)

fpga.write('VCM1_RE_BRAM',struct.pack('>1l',((c2amp[2*j]+c2amp[(2*j)+1])/2
)*cos((pi/180)*((c2phase[2*j]+c2phase[(2*j)+1])/2))*2**24),4*j)  #ch 1,5,9
... (4i+1) (interpolated)

fpga.write('VCM1_IM_BRAM',struct.pack('>1l',((c2amp[2*j]+c2amp[(2*j)+1])/2
)*sin((pi/180)*((c2phase[2*j]+c2phase[(2*j)+1])/2))*2**24),4*j)

fpga.write('VCM2_RE_BRAM',struct.pack('>1l',c2amp[(2*j)+1]*cos((pi/180)*c2
phase[(2*j)+1])*2**24),4*j)  #canal 2,6,10 ... (4i+2)

fpga.write('VCM2_IM_BRAM',struct.pack('>1l',c2amp[(2*j)+1]*sin((pi/180)*c2
phase[(2*j)+1])*2**24),4*j)
```

```
fpga.write('VCM3_RE_BRAM',struct.pack('>1l',((c2amp[(2*j)+1]+c2amp[(2*j)+2
])/2)*cos((pi/180)*((c2phase[(2*j)+1]+c2phase[(2*j)+2])/2))*2**24),4*j)
#canal 3,7,11 ... (4i+3) (interpolated)

fpga.write('VCM3_IM_BRAM',struct.pack('>1l',((c2amp[(2*j)+1]+c2amp[(2*j)+2
])/2)*sin((pi/180)*((c2phase[(2*j)+1]+c2phase[(2*j)+2])/2))*2**24),4*j)
########################## writing C3 #######################

fpga.write('VCM4_RE_BRAM',struct.pack('>1l',c3amp[2*j]*cos((pi/180)*c3phas
e[2*j])*2**24),4*j)

fpga.write('VCM4_IM_BRAM',struct.pack('>1l',c3amp[2*j]*sin((pi/180)*c3phas
e[2*j])*2**24),4*j)

fpga.write('VCM5_RE_BRAM',struct.pack('>1l',((c3amp[2*j]+c3amp[(2*j)+1])/2
)*cos((pi/180)*((c3phase[2*j]+c3phase[(2*j)+1])/2))*2**24),4*j)

fpga.write('VCM5_IM_BRAM',struct.pack('>1l',((c3amp[2*j]+c3amp[(2*j)+1])/2
)*sin((pi/180)*((c3phase[2*j]+c3phase[(2*j)+1])/2))*2**24),4*j)

fpga.write('VCM6_RE_BRAM',struct.pack('>1l',c3amp[(2*j)+1]*cos((pi/180)*c3
phase[(2*j)+1])*2**24),4*j)

fpga.write('VCM6_IM_BRAM',struct.pack('>1l',c3amp[(2*j)+1]*sin((pi/180)*c3
phase[(2*j)+1])*2**24),4*j)

fpga.write('VCM7_RE_BRAM',struct.pack('>1l',((c3amp[(2*j)+1]+c3amp[(2*j)+2
])/2)*cos((pi/180)*((c3phase[(2*j)+1]+c3phase[(2*j)+2])/2))*2**24),4*j)

fpga.write('VCM7_IM_BRAM',struct.pack('>1l',((c3amp[(2*j)+1]+c3amp[(2*j)+2
])/2)*sin((pi/180)*((c3phase[(2*j)+1]+c3phase[(2*j)+2])/2))*2**24),4*j)

    elif cal_mode == 2:  # Ideal IF Hybrid camp=1, cphase=-90
        for i in range(512):
            print ('Writing coeficient '+str(i)+' of 510'+' please
wait...')
            fpga.write('VCM0_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM0_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM1_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM1_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM2_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM2_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM3_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM3_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM4_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM4_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM5_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM5_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM6_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM6_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)
            fpga.write('VCM7_RE_BRAM',struct.pack('>1l',+0*2**24),4*i)
            fpga.write('VCM7_IM_BRAM',struct.pack('>1l',-1*2**24),4*i)

    print('--- Calibration completed  ---')

################## Calibration completed  #########################
```

```
    modo=1   # Choosing type of plot

#################  Ploting SSR (MODO 1) #########
    if modo == 1:
        srr_datos.write('#SRR_values    #Frequency   \n')
        for i in range(points):
            print('meauring RF = '+str(rf[i])+' GHz')
            rys.write("freq " + str(rf[i]) + "ghz\r\n")
            time.sleep(0.5)
            acc_n, interleave_log_i, interleave_log_q = get_data()
            interleave_log_i[0]=interleave_log_i[1] #fixing DC offset
            interleave_log_q[0]=interleave_log_q[1] #fixing DC offset
# Plots
            g0.plot(interleave_log_i)
            g1.plot(interleave_log_q)
# Calculates sideband rejection
            if rf[i]<LO:
                srr.append([rf[i],max(interleave_log_q)-
max(interleave_log_i)])
            elif rf[i]>=LO:
                srr.append([rf[i],max(interleave_log_i)-
max(interleave_log_q)])

############### Plots and writes SRR ###########
            g2.plot(srr)
            srr_datos.write(str(srr[i][1])+' '+str(rf[i])+' \n')
#
    elif modo ==2:
######### INICIO DE TONO DE PRUEBA (LSB AND USB SPECTRA FOR TEST TONE)
(MODO 2) #####
        agi.write("freq " + str(2.6) + "ghz\r\n")   # TONO DE PRUEBA  en
2600 MHz
        time.sleep(0.5)
        acc_n, interleave_log_i, interleave_log_q = get_data()
        time.sleep(0.1)

        g0.plot(interleave_log_i)
        g1.plot(interleave_log_q)

        for i in range(0,len(interleave_log_b)):
            lsb_usb_testtone.write(str(interleave_log_i[i])+'
'+str(interleave_log_q[i])+' '+str(i)+' \n')

    ######################################################################
    ######################################################################
    lsb_usb_testtone.close()
    t_ejec = time.time()-t1
    hh,mm,ss=hhmmss(t_ejec)
    print('done in: '+str(trunc(hh))+'[h] ,'+str(trunc(mm))+'[min]
,'+str(trunc(ss)),'[seg].')
    srr_datos.write('# FIN de datos SRR \n')
    srr_datos.close()
    print ('')
    print ('SRR measurement finished!  - '+str(points)+' points')
    print ('')
```

```
####  this is to continue plotting the USB and LSB spectra ####
    ok=1
    while ok==1 :
        acc_n, interleave_log_i, interleave_log_q = get_data()
        time.sleep(0.5)
        g0.plot(interleave_log_i)
        g1.plot(interleave_log_q)

######### This code is to save each run with a different archive name
#######
#    out=raw_input('Enter a filename to save data:(already saved to
srr_data.dat)')
#    if out!='':
#        f = open(str(out), 'w')
#        f.write("RF (GHz),SSR (dB)\n")
#        for i in range (points):
#            f.write(str(srr[i][0])+","+str(srr[i][1])+"\n")

except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()

exit_clean()
```

Appendix B: List of Publications

Articles

A calibrated digital sideband separating spectrometer for radio astronomy applications, R. Finger, P. Mena, N. Reyes, R. Rodriguez, L. Bronfman. Publications of the Astronomical Society of the Pacific, Vol. 125, No. 925, pp. 263-269 (March, 2013).

Conferences

R. Rodriguez , R. Finger, P. Vasquez ,R. Bustos , N. Reyes b , P. Zorzi , L. Bronfman, F.P. Mena, “New capabilities for the Southern 1.2-m mm-Wave Telescope" in Millimeter and Submillimeter Detectors and Instrumentation for Astronomy V, edited by Wayne S. Holland; Jonas Zmuidzinas, Amsterdam, July 2012.

R. Finger, P. Mena, N. Reyes, R. Rodriguez, L. Bronfman, “A calibrated digital sideband separating spectrometer for radio astronomy applications”, The 24th International Symposium on Space Terahertz Technology, ISSTT 2013, Groningen, the Netherlands.

Bibliography

A. Surkov, et al. (2006). *94 GHz Radar Sensor for Process Control and Imaging.* ECNDT.

ALMA. (2013, March 20). ALMA system block diagram. EDM document number: SYSE-80.04.01.00-004-Q-DWG. Retrieved from edm.alma.cl

Axel Murk, et al. (2009). Characterization of a 340 GHz Sub-Harmonic IQ Mixer with Digital Sideband Separating Backend. *5th ESA Workshop on Millimetre Wave Technology and Applications & 31st ESA Antenna Workshop.* Noordwijk.

B. Henderson and J.Cook. (1985). *Image-Reject and Single-Sideband Mixers.* Retrieved March 20, 2013, from http://www.triquint.com/products/tech-library/docs/WJ/ImageRej_n_SSB_mixers.pdf

Bin Zhou, et al. (2009). Pipeline FFT Architectures Optimized for FPGAs. *International Journal of Reconfigurable Computing*.

D'Addario, L and Thompson, A ,(2000). *ALMA Memo 304, Relative Sensitivity of Double- and Single-Sideband Systems for both Total Power and Interferometry.*

F. P. Mena, et al. (2011). Design and Performance of a 600–720-GHz Sideband Separating Receiver Using AlOx and AlN SIS Junctions. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES*, Vol.59.

Heymsfield, A. et al. (2008). The 94-GHz radar dim band: Relevance to ice cloud properties and CloudSat. *GEOPHYSICAL RESEARCH LETTERS*, Vol.35.

J. R. Fisher and M. A Morgan. (2008). *Analysis of a Single-Conversion, Analog/Digital Sideband-Separating Mixer Prototype.* National Radio Astronomy Observatory Electronics Division Internal Report No. 320 .

Koyel Das, et al. (2011). Conversion from linear to circular polarization in FPGA. *Astronomy & Astrophysics*.

Lucas, R. (2000). *ALMA Memo #300. "Reducing Atmospheric Noise in Single Dish observations with ALMA".*

M. Morgan and J. R Fisher. (2010). Experiments With Digital Sideband-Separating Downconversion. *Publications of the Astronomical Society of the Pacific*, vol. 122, no. 889, pp. 326-335.

Ma, H. (2011). *Band 3 Cartridge Acceptance Report (SN 60) ALMA-EDM Document Number: FEND-40.02.03.00-0573-A-REP.*

Mahieu, S. (2011). Status of ALMA Band 7 Cartridge Production. *22nd International Symposium on Space Terahertz Technology.* Tucson.

Mangum, P. R. (1998). System Temperatures, Single Versus Double Sideband Operation, and Optimum Receiver Performance. *International Journal of Infrared and Millimeter Waves* .

R.P. Millenaar and A.-J. Boonstra. (2011). *The radio frequency interference enviroment at candidate SKA sites.* SKA.

Reyes, N. (2013). *Design of a receiver at 31-45 GHz based on HEMT amplifiers and Shottky mixers.* Santiago: Universidad de Chile.

Thompson, A. R. (1986). *Interferometry and Synthesis in Radio Astronomy.* New York: Wiley.

Vilaro, B. (2011). *ALMA Cycle 0 Technical Handbook.* http://almascience.eso.org/documents-and-tools/cycle-0/alma-technical-handbook/view.

Wallace, H. B. (2010). Analysis of RF imaging applications at frequencies over 100GHz. *Applied Optics*, vol.49.