

Technische Universität Wien

Master Thesis

**Multicarrier Transmission Gain and Group
Delay Measurements**

Author:

Arash Jafari

Advisor:

Univ. Prof Christoph Mecklenbräuer

Date of issue: 2019-11-25

Number of pages: 123



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

This thesis documents the Multi-Carrier Group Delay and gain measurements for OneWeb satellites with frequency translation in satellite communication. The uplink is in the Ka-band and the downlink is in the Ku-band. The transmission gain and absolute group delay measurements are implemented although the local oscillator (LO) is not accessible. These measurements are implemented by using the OneWeb Payload Test System for Radio Frequency Special Check-Out Equipment Electrical Ground Support Equipment (PTS RF SCOE EGSE). The PTS RF SCOE consists of distinct measurement units for various measurement goals. The OneWeb RF SCOE consists of a signal generator, spectrum analyzer, switch control unit, RF matching unit, RF FrontEnd and a host server to link these components together. For more information about RF SCOE please refer to <https://atos.net/en/products/aerospace-defense-electronics/egse>.

The focus of this document is on multi-carrier gain and group delay measurement of the OneWeb satellite which henceforth will be called the Device Under Test (DUT). The two main measurement units that are utilized to perform the measurements for transmission gain and group delay are two signal generators and a spectrum analyzer.

Having information about the phase and magnitude response of the DUT is sufficient to measure the gain and group delay of the DUT. The Rohde and Schwarz FSW 43 spectrum analyzer includes an I/Q-Analyzer. The magnitude and phase of the input RF signal are measured by reading I/Q-data from the spectrum analyzer. Consequently, the measured magnitude and phase are calculated to measure the group delay and gain of DUT with frequency translation.

This thesis provides information on how to obtain the necessary parameters from Analog Spectrum Analyzer (ASA) for measuring the transmission gain and absolute group delay. Measuring the gain and group delay need two specific calibrations prior to the multi-carrier transmission gain and absolute group delay measurement. The TAC and multi-carrier calibrations give us the reference gain and phase response versus frequency curves for Multi-Carrier Measurements (MCMs). All specific calculations are performed for accurately estimating the absolute group delay and transmission gain of the DUT. Three possible fast internal calibrations are suggested as a replacement of the repetitive and time-consuming TAC and multi-carrier calibrations. Finally, measuring the transmission gain of the same DUT with different interfaces of the RF front end ends up to constant transmission gain of the DUT.

Table of Contents

Chapter 1 Introduction.....	2
1.1 Background.....	2
1.2 Objective and Scope	3
Chapter 2 K17 alternative.....	7
2.1 K17 Introduction.....	7
2.2 IQ-Analyzer introduction.....	8
2.2.1 Unambiguous group delay results.....	9
2.2.2 Sample Rate	10
2.2.3 Analysis Bandwidth.....	10
2.2.4 Maximum Bandwidth	10
2.2.5 Meas Time	10
2.2.6 Record Length.....	10
2.2.7 RBW	11
2.2.8 Sweep Points.....	11
2.2.9 K17 functionality implemented in TCL.....	11
2.2.10 Defining Record Length and Sample Rate:	11
2.2.11 Query Trigger Offset.....	15
2.2.12 Reading IQ data from spectrum analyzer:	16
2.2.13 Averaging and Fast Fourier Transform (FFT)	16
2.2.14 First and second method comparison.....	19
2.2.15 Magnitude and phase calculation.....	22
2.2.16 Magnitude verification.....	22
2.2.17 Phase verification.....	24
2.2.18 Slope correction	26
2.3 K17 option and IQ-Analyzer comparison.....	29
2.3.1 Different spacing.....	29
Chapter 3 Measurement prerequisites.....	36
3.1 Payload test system overview	36
3.2 Multicarrier waveform.....	37
3.2.1 Measure transmission gain and group delay with Multicarrier stimulus	38
Chapter 4 Turn Around Converter Calibrations	40
4.1.1 Gain calibration.....	41
4.1.2 Phase Calibration	42
Chapter 5 Multicarrier Calibrations.....	44
5.1 Calibration of the arbitrary wave generator flatness CAL_IQ.....	44
5.2 Multicarrier calibrations (CAL_MCARRIER)	45
5.2.1 Configurable parameters of the multicarrier setting window.....	46

5.2.2 Multicarrier calibration routine	48
5.2.3 Multicarrier amplitude calibrations.....	49
5.2.4 Multicarrier phase calibrations.....	49
Chapter 6 Multicarrier measurements	52
6.1 Configurable parameters of MEAS_MCARRIER.....	52
6.2 Results of multi-carrier measurements	55
6.3 Multi-carrier measurements routine.....	55
6.3.1 Gain (magnitude flatness) measurements	56
6.3.2 Group delay measurements	56
6.3.3 Measurement Results	58
6.3.4 splitSpan.....	59
6.3.5 Stimulus and response expansion	61
6.3.6 Trace smoothing and remove outlier	64
6.3.7 Eliminating DC offset and IQ imbalance.....	67
6.3.8 Multicarrier gain calibration using internal loop	69
6.4 Impact of different chirp carrier spacings and input power levels on the group delay measurements	73
Chapter 7 Transmission gain ripple compensation	74
7.1 Transmission gain ripple compensation for uplink-side in the original setup	80
7.2 Ripple compensation downlink-side return loss degradation	82
7.3 Ripple compensation two-sided by return loss degradation.....	84
Chapter 8 Conclusions and future work.....	86
Bibliography	88
Appendix I	89
Appendix II.....	94
Appendix III	114

Abbreviations, Terms, and Definitions

The following Acronyms and Abbreviations are used in this document:

A		Ka	K above
ALC	Automatic Latch Control	Ku	K under
AOS	Airbus Oneweb satellite	L	
ASA	Analog Spectrum Analyzer	LAN	Local Area Network
AWG	Arbitrary Waveform Generator	LHCP	Left Hand Circular Polarization
B		LNA	Low Noise Amplifier
bps	bits per second	LO	Local Oscillator
C		M	
CCS	Central Checkout System	MC	Multi Carrier
D		MCC	Multi Carrier Calibration
DAC	Digital Analog Converter	MCM	Multi Carrier Measurement
dB	decibel	MCGD	Multi Carrier Group Delay
dBc	dB related to center frequency	Mbps	Mega bit per second
dBm	dB related to 1 mW	O	
DC	Direct Current	P	
DUT	Device Under Test	PCB	Printed Circuit Board
DL	DownLink	PM	Phase Modulation
E		PTS	Payload Test System
ECAL	Electronic CALibration	PWM	PoWer Meter
ECSS	European Cooperation for Space Standardization	PWS	PoWer Sensor
EDRS	European Data Relay System	Q	
EGSE	Electrical Ground Support Equipment	QPSK	Quadrature Phase Shift Keying
F		R	
FFT	Fast Fourier Transform	RF	Radio Frequency
FEE	Front End Equipment	RHCP	Right Hand Circular Polarization
G		RX	Receiver
GSE	Ground Support Equipment	RFTS	Radio Frequency Test System
GUI	Graphical User Interface	R&S	Rohde & Schwarz
I		RBW	Resolution BandWidth
IQ	In-phase and Quadrature	S	
K		SAN	Signal and spectrum ANalyser
kBps	210 Bits per second	S/C	SpaceCraft
		SCOE	Special Check-Out Equipment
		SSPA	Solid State Power Amplifier

T

TAC Turn Around Converter

TCL Tool Command Language

TPIS Trigger Point In Sample

TTC Telemetry and TeleCommand

U

UL UpLink

V

VSG Vector Signal Generator

VNA Vector Signal Analyzer

Chapter 1

Introduction

1.1 Background

Satellite-based internet access provides a wireless connection involving three satellite dishes; one at the internet service provider's hub, the second one in space and the third one attached to the user's premises. In addition to the satellite dish, you also need a modem and coaxial lines running to and from the dish to your modem.

OneWeb, formerly known as WorldVu Satellites, is a global communications company founded by Greg Wyler. The company is headquartered in London, United Kingdom, and was founded in Arlington (VA), USA. It commenced launches of the OneWeb satellite constellation, a network of more than 650 LEO satellites, on February 27, 2019.

Its intended goal is to provide internet services to "everyone, everywhere" delivering much-needed connectivity to rural and remote places as well as to a range of markets including aero, maritime, land mobility, cellular backhaul. The OneWeb constellation started with an initial 650-satellite constellation and currently being built out to provide global satellite internet broadband services to people everywhere starting in 2021.

The classification group of the OneWeb satellite is a small satellite with a weight of around 175-200 Kg. The satellites will operate in circular LEO, at approximately 750 miles (1,200 km) altitude, transmitting and receiving in the Ku and Ka-band of the radio frequency spectrum.

The SCOE, with the Overall Check-Out Equipment (CCS), constitutes the EGSE subsystem and allows performing Payload and TTC RF testing of the platform on the ground during assembly, integration, and tests, on a complete satellite. The RF SCOE (so-called PTS) is an SCOE that allows performing RF tests on the OneWeb payload and on the TTC subsystem.

1.2 Objective and Scope

Since the frequency spectrum is a limited resource. There is continual pressure to utilize this resource more efficiently. The end result is that transmitters and receivers must accommodate higher-order digital modulation techniques in order to accommodate higher data rates in limited bandwidths. This leads to more effective sharing of the frequency spectrum, and also higher data rates and quality of service options for the end customer. Ideal networks and components pass modulated signals containing information without producing any additional distortion. They are characterized by both flat magnitude and linear phase shift (constant slope) over frequency. It implies that all the frequency components of a signal passing through an ideal network will experience the same amount of time delay. However, in a non-ideal network, distortion is created by the amplitude and phase non-linearities present in the device. Any variation in delay or amplitude transmission results in signal distortion[1].

Therefore, the characterization of signal delay for all frequencies of interest by performing suitable measurements to acquire the quality of the signal transmission of the RF components is useful for the FR engineer. RF engineers always use group delay for measuring the phase nonlinearity characteristic of the RF components. Similarly, the transmission gain can be used for measuring the variations in the magnitude of the input signal as it passes through RF components.

OneWeb satellite has increased the information bandwidth of the satellite and utilize modulation schemes. Hence, the non-linearity of the system components becomes an important issue, deviation from linear phase and calculating the group delay can be measured to completely characterize a device. The slope of the phase ripple is dependent on the number of ripples that occur over the unit of the frequency range. Since the group delay is the differentiated phase response, it takes the number of ripples into account. The group delay is often a more easily interpreted indication of phase distortion[2].

Typically, the group delay of a non-FTD is measured simply with a network analyzer. The network analyzer Measures the group delay by sweeping over the bandwidth of interest and measuring point by point the amplitude and phase of the signal.

Frequency-translation devices (FTDs) such as mixers, converters, and tuners are critical components in most RF and microwave communication systems. As communication systems adopt more advanced types of modulation, FTD designs are increasingly complex, tests are more stringent with tighter specifications, and the need to reduce costs is more important than ever[3]. The measuring group delay of FTDs has more complexity compared to non-FTDs because the input signal frequency components of the DUT is different than output signal frequency components. The measurement trade-offs for frequency-translating devices vary widely among different industries. Measurement accuracy, speed, cost and ease of setup are among the considerations for determining the best test equipment.

In the past, measurements of FTDs using the network analyzer were limited to relative group delay and conversion loss relative to a golden DUT. The measurement of absolute group delay was limited to alternate techniques utilizing a low-frequency signal modulated on the high-frequency carrier. The signal is demodulated, and the group delay is measured. This technique requires additional test equipment and is resolution limited to around 10 ns. However, there is a technique to measure the absolute group delay of FTDs by utilizing a fully characterized mixer calibration standard and implementing vector error correction on the VNA measurements[4]. This method has a drawback that a common LO is needed for calibration mixer and FTD in order to measure the absolute group delay measurement.

If the DUT implements frequency conversion, there are two cases need to be taken into consideration for measuring the absolute group delay:

1- FTDs with accessible LO:

The common technique for phase and group delay measurement on FTD with accessible LO uses a reference mixer in the reference path of the VNA. This reference mixer employs the same LO as the DUT to reconvert the frequency of the reference signal to the frequency of the RF or IF measurement signal. This method eliminates the influence of the LO phase and frequency instabilities.

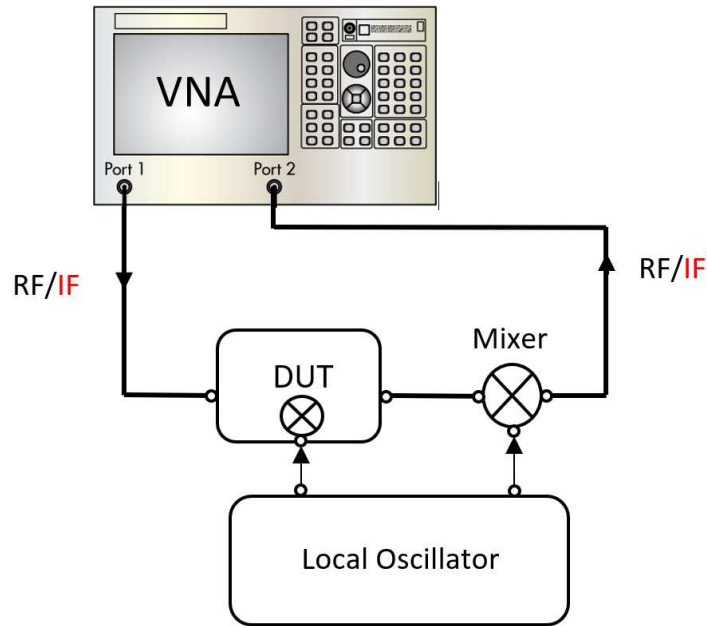


Figure 1: Group delay measurement of the FTD with a common LO.

2- FTDs with non-accessible LO:

However, there is a new technique to measure the group delay of a frequency conversion DUT using the embedded local oscillator in case the local oscillator of the DUT is not accessible. For example, the embedded local oscillator option of the Keysight NWA uses this new technique. An external signal generator with 10MHz reference and GPIB inputs is used as a substitution for the LO during calibration and measurement. The key aspect is the tracking frequency of the IF of the DUT such that the frequency of the external LO used for the reference channel mixer can be adjusted to accommodate an offset and drift in the DUT embedded LO. Furthermore, the phase of the IF of the DUT is also tracked to accommodate phase shift or slight frequency error (less than 1 Hz offset is required to avoid difficulties in the delay measurement). This frequency tracking is done through software techniques that do not require additional phase-locking hardware. The measurement accuracy depends on the frequency and phase stability of the external signal generator's LO and overall noise in the measurement. Figure 2 shows the group delay measurement setup of the frequency conversion DUT with the help of an external mixer.

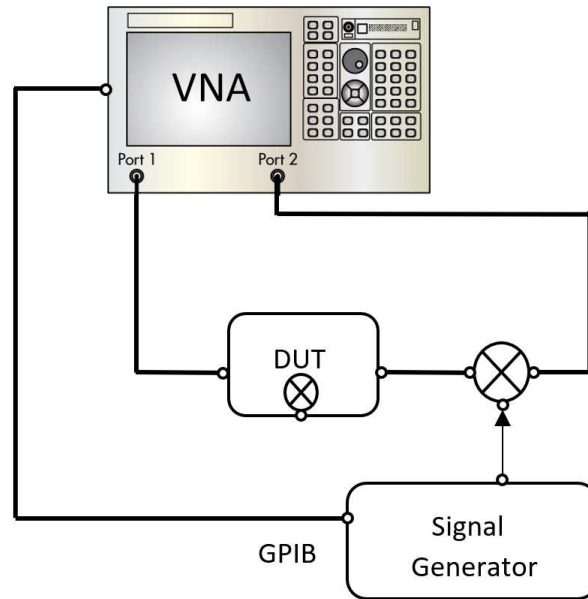


Figure 2: Group delay measurement setup of frequency conversion DUT with the help of an external mixer.

The most common tool for characterizing non-frequency translating components is the vector network analyzer (VNA). Due to their design and error correction capabilities, they are very fast and accurate FTDs, such as mixers, are more difficult to characterize due to the frequency offset between input and output, and hence cannot be measured by a VNA alone. Up to now, all techniques use test FTD's so that the frequency at the reference and test channels is the same. In this way, VNA's can be used to characterize FTD's with accuracies near those obtained for non-FTD's. The most common FTD measurement technique uses a network analyzer and a reference test mixer to obtain the amplitude and phase match between FTD's[5]. This reference test mixer should be fed by the same LO as the FTD. The disadvantage of this approach is that the accuracy will always be limited to how well the standard has been characterized. This method also fails to measure the transmission gain and group delay of the FTDs with non-accessible LO.

In the satellite communication or more specifically in OneWeb PTS SCOE, the FTD is usually the satellite with a non-accessible LO. In this thesis, For the reason of costly network analyzer and complex measurement setup, the transmission gain and group delay of the satellite is measured over a wide frequency range with two external signal generators and a spectrum analyzer.

There are many variations for methods of measuring group delay and transmission gain. One of these methods stimulates the device under test by an MC signal and measures the phase difference of the output relative to the stimulus signal. In this method, we measure the phase shifts of multiple carriers in the stimulus signal across the frequency band of interest.

This thesis presents the techniques that can be used to characterize the amplitude and phase response of the FTD such as satellite with non-accessible LO. This is done by accurately measuring the transmission gain and the absolute group delay. Frequency-translating devices present unique measurement challenges since their input and output frequencies differ[3].

In the OneWeb satellite, eight channels utilize the Ka frequency band to transmit data. The payload test system has also the possibility to measure the same band of the frequency with different interfaces of the front end simultaneously. Measuring the transmission gain of the satellite with different interfaces should end in the

same measurement result. The impedance mismatches between RF front end interfaces and the satellite cause and error in the transmission gain measurement of the satellite.

This thesis seeks to find the answer for the following research questions:

1. How to measure the transmission gain of the satellite which performs frequency conversion with non-accessible LO.
2. How to measure the absolute group delay of the satellite which performs frequency translation with non-accessible LO.
3. Improve the result of transmission gain measurement errors caused by impedance mismatches of the RF front end.

To get the most from this thesis, you should have a basic understanding of frequency translation terminology, such as RF, IF, and LO ports. Understanding of fundamental RF, analog spectrum analyzer, arbitrary wave generator, network analyzer terms such as S-parameters, group delay, transmission gain, match, and RF calibration is also expected.

Chapter 2

K17 alternative

2.1 K17 Introduction

The R&S FSW-K17 option is a firmware application that adds functionality to the R&S FSW for Multi-Carrier Group Delay measurements.

The Multi-Carrier Group Delay application features:

- Highly accurate group delay measurement for large spans
- Orthogonal measurement method
- Group delay measurement with frequency conversion.
- Storage and loading functions for reference data
- Storage functions for measurement settings and results
- Graphical display of group delay, gain, magnitude, and phase at carrier frequency points in measurement or reference signal, and phase difference between measurement and the reference signal.

Unfortunately, the Multi-Carrier Group Delay measurement as provided by the K17 option needs much improvement because the K17 option does not work satisfactorily when the user intends to measure the absolute group delay for a DUT including a mixer which implements a frequency conversion.

The concept of the K17 option is to evaluate the output signal from the DUT which is stimulated by a vector signal generator. After a preliminary reference measurement of the stimulation signal without the DUT, the output signal from the stimulated DUT is measured and subsequently evaluated.

Instead of evaluating the timing of the signals, the phase shift and magnitude response at multiple carriers across the frequency band of interest is measured. A baseband stimulus signal consisting of several unmodulated carriers (in our case chirp signal) with a fixed inter-carrier spacing is used as the stimulus signal, allowing for a very fast wideband measurement. By measuring the phase differences between the two signals at the input and at the output of the DUT, the K17 option application calculates the relative phase between output and input.

The phase and magnitude of the carriers at the input of the DUT have to be measured and stored as the reference curve. In this way, no reference path or connection is needed between the input and the output of the DUT. A reference mixer in the signal generator provides a phase reference at the IF frequency. Thus, a constant delay factor is eliminated, and the group delay is calculated relative to the phase reference[6].

Figure 4 shows the recommended calibration and measurement setup for MCGD measurement using the K17 option for FSW ASA. The DUT in this setup is considered as a non-frequency converting device. The SMx is a signal generator manufactured by R&S. An SMW200A AWG is used instead of the SMx signal generator in the OneWeb RF SCOE. A synchronization cable for the common 10 MHz reference signal is missing in Figure 3.

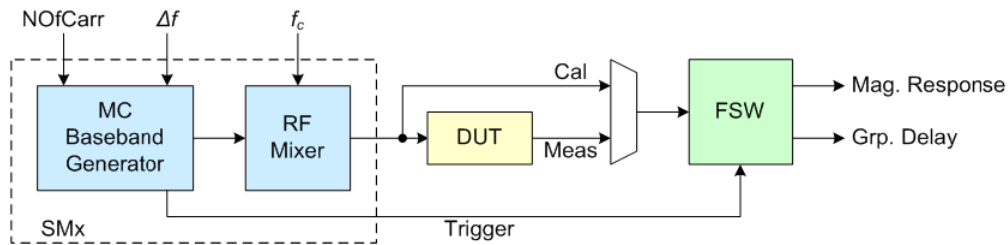


Figure 4: MCGD measurement setup for the K17 option [6].

Where:

NOfCarr: Number of carriers.

Δf : Carrier spacing of the multicarrier stimulus.

f_c : Center frequency of the multicarrier stimulus.

2.2 IQ-Analyzer introduction

The R&S FSW I/Q-Analyzer is a firmware application that adds functionality to perform I/Q data acquisition and analysis to the R&S FSW. The R&S FSW I/Q Analyzer application is part of the standard base unit and requires no further installation. The main R&S FSW I/Q-Analyzer features are the acquisition of analog I/Q data and the export of the I/Q data to the other application.

2.2.1 Unambiguous group delay results

In the time domain, a multi-carrier signal with a period of $1/\langle\text{CarrierSpacing}\rangle$ simply repeats itself. The R&S FSW MCGD application cannot distinguish the two signals. Therefore, any group delays larger than $1/\langle\text{CarrierSpacing}\rangle$ are no longer unambiguous.

The range in which unambiguous group delay results can be measured is defined by the following equation:

$$\text{Unambiguous Range} = \pm \frac{1}{2 \times \text{Carrier spacing}}$$

Equation (1): Unambiguous range of group delay measurements.

Any group delay that exceeds this range is wrapped back into the unambiguous range. A chirp signal with 0.1 MHz carrier spacing is used as a stimulus which has $\pm 5 \mu\text{s}$ unambiguous range. In order to avoid ambiguous results, the carrier spacing is defined such that the expected group delay does not exceed the unambiguous range.

For the absolute group delay measurement, an external trigger is required to achieve sufficient accuracy for the phase measurement because the AWG which generates the stimulus signal for the calibration and measurement is not synchronized to the ASA[7].

2.2.1.1 Trigger Uncertainty

For the absolute group delay measurement using an external trigger, the trigger uncertainty in FSW is always 5 ns, because the sample rate of the trigger line is 200 MHz but the FSW samples data internally at a much higher rate and the trigger uncertainty is corrected and can be queried. The trigger uncertainty value is queried and considered in the group delay calculation. In the I/Q Analyzer application, only "External Trigger 1" is supported[7].

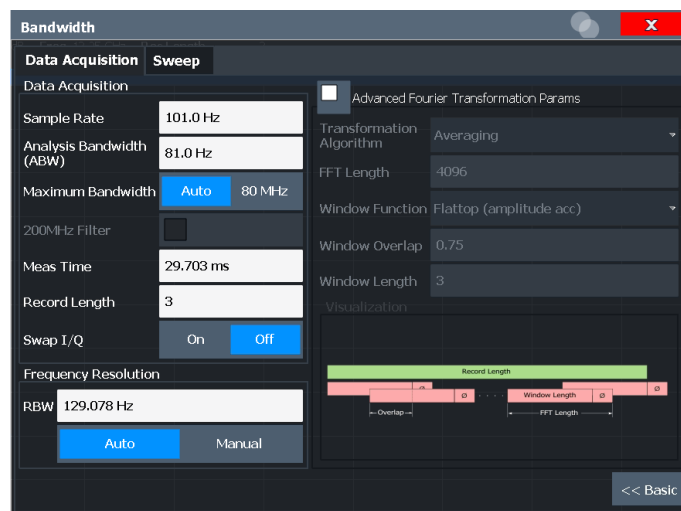


Figure 5: IQ Analyser data acquisition configuration window

2.2.2 Sample Rate

The Sample Rate parameter defines the I/Q data sample rate of the R&S FSW. This value is dependent on the defined analysis bandwidth and the defined signal source. Up to the maximum bandwidth of 80 MHz, the following rule is followed by the I/Q Analyzer[7].

$$\text{Sample rate} = \frac{\text{Analysis bandwidth}}{0.8}$$

Equation (2): Sample rate and analysis bandwidth relation.

2.2.3 Analysis Bandwidth

The Analysis Bandwidth parameter defines the flat, usable measurement bandwidth of the acquired I/Q data. This value is dependent on the defined sample rate and the defined signal source[7].

Up to the maximum bandwidth, the following rule provided by R&S applies:

$$\text{Analysis bandwidth} = 0.8 \times \text{Sample rate}$$

Equation (3): The relation between sample rate and analysis bandwidth in R&S FSW 43.

2.2.4 Maximum Bandwidth

The Maximum Bandwidth defines the maximum bandwidth to be used by the R&S FSW for I/Q data acquisition. This setting is only available if the bandwidth extension option R&S FSW-B160/-B320/- B500 is installed. Otherwise, the maximum bandwidth is determined automatically[7].

2.2.5 Meas Time

The measurement time (Meas Time) Defines the I/Q acquisition time. By default, the measurement time is calculated as the number of I/Q samples ("Record Length") divided by the sample rate. If one changes the measurement time, the Record Length changes accordingly[7].

2.2.6 Record Length

The Record Length defines the number of I/Q samples to record. By default, the number of sweep points is used. The record length is calculated as the measurement time multiplied by the sample rate. If you change the record length, the **Meas Time** changes accordingly[7]. In Figure 5 the record length is calculated as the multiplication of the sample rate and the **Meas Time**.

$$\text{Record Length} = \text{Meas Time} \times \text{Sample Rate}$$

Equation (4): Record length calculation.

2.2.7 RBW

The RBW (Resolution BandWidth) defines the resolution bandwidth for time-domain signal results. The available RBW values depend on the sample rate and record length. Depending on the selected RBW mode, the RBW value is either determined automatically or can be defined manually[7].

2.2.8 Sweep Points

In the I/Q Analyzer application, specific frequency bandwidth is swept for the specified measurement time. During this time, a defined number of samples (= "Record Length") are captured. These samples are then evaluated by the IQ analyzer applications. Therefore, the number of sweep points does not define the amount of data to be acquired, but rather the number of tracepoints that are evaluated and displayed in the result diagrams[7].

2.2.9 K17 functionality implemented in TCL

In this Master thesis, the needed functionality of the R&S K17 option is implemented in the TCL language for the purpose of accurate group delay measurements. Here <https://www.tcl.tk/man/> you can find more information about TCL/TK language.

2.2.10 Defining Record Length and Sample Rate:

For processing the IQ data in the time domain and transforming them into the frequency domain, the user needs to define the sample rate and record length in the correct way. Selecting a wrong record length or sample rate leads to incorrect magnitude and phase measurement results. In the following, the correct definition of sample rate and record length is explained.

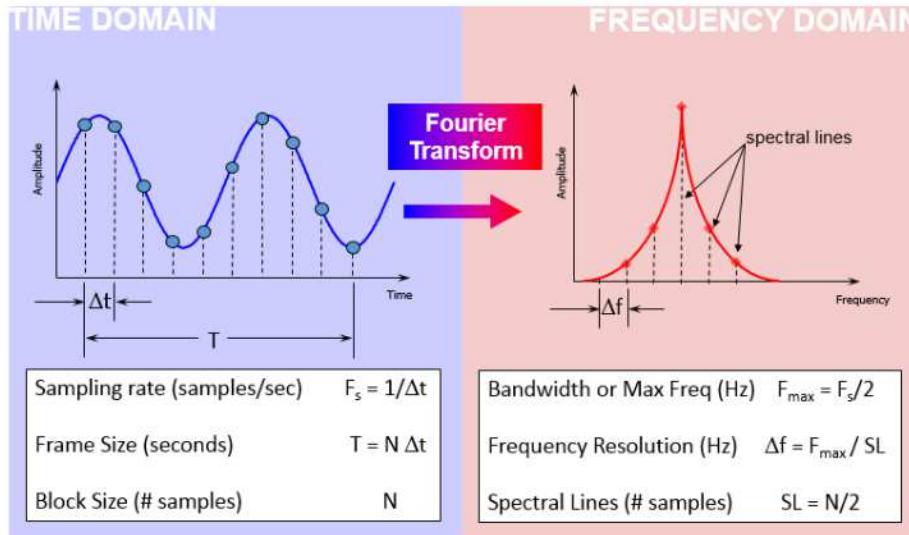


Figure 6: Sample analysis in time-frequency conversion[8].

There are three important terms on both time and frequency domain representation that can affect the quality of the final analysis.

- Sampling rate (F_s) is the number of IQ data samples acquired per second.
- Frame Size (T) is the amount of time that data is collected to perform a Fourier transform.
- Block Size (N) is the total number of data samples acquired during one frame which is an even integer number. There is a force

$$F_s = \frac{1}{\Delta t} = \frac{1}{T_s}$$

Equation (5): Sampling rate calculated from the sampling period.

$$T = N_{initial} \times \Delta t = \frac{N_{initial}}{F_s} = \frac{1}{\text{spacing of chirp signal}}$$

Equation (6): The frame size is reverse of carrier spacing.

$$N_{initial} = \frac{F_s}{\text{spacing of chirp signal}} = F_s \times T$$

Equation (7): Initial block size calculation.

Where:

T_s : Sampling period

T : Frame Size (measurement time in data acquisition)

F_s : Analysis bandwidth

$N_{initial}$: Block size (record length or number of samples)

The sample rate is specified through the bandwidth of the chirp signal which is used as the stimulus (Analysis Bandwidth in data acquisition of IQ-Analyzer). The key point is that the result of the analysis bandwidth divided by the carrier spacing of the chirp stimulus has to be a natural number. The analysis bandwidth (F_s) has to be exactly a multiple natural numbers of the carrier spacing[9].

The `ASA1.getdata_mcsignals_iq` is a procedure implemented in TCL language to query data from the time-domain signals at IQ-Analyzer of the ASA. The user must define the center frequency, span, carrier spacing of the chirp signal as inputs of the procedure. All input parameters are configurable from the RF SCOE GUI. The outputs of the procedure are the frequency list, magnitude list, and phase list. For more information, please check Appendix I.

$$USR = \frac{span}{0.8}$$

Equation (8): The relation between the uncorrected sample rate and span of the measured signal.

$$URL = \frac{USR}{spacing}$$

Equation (9): Calculation of uncorrected record length from uncorrected sample rate and spacing of the multi-carrier signal.

$$correction\ factor = spacing \times (1 - (URL - integer(URL)))$$

Equation (10): Correction factor calculation.

$$CSR = USR + correction\ factor$$

Equation (11): Correct sample rate calculation.

$$IRL = integer(rounded(\frac{CSR}{spacing}))$$

Equation (12): Initial record length is sample rate over spacing.

Where:

$span$: MC chirp stimulus span (less than 80 MHz)

$spacing$ = Spacing of MC chirp stimulus (usually considered 0.1 MHz)

USR: Uncorrected Sample Rate

URL: Uncorrected Record Length

CSR: Corrected Sample Rate

IRL: Initial Record Length

First, an initial record length ($N_{initial}$) is defined which delivers the whole signal but only one period in the time domain. Afterward, it is multiplied with our intended number of periods in order to achieve a good estimation of the signal. Each period represents the whole signal, actually, by acquiring more than one period of the signal in the time domain, we are applying at the same time averaging of the signal[9].

The number of queried periods of the signal is hard limited by the capabilities of our software and the ASA to read the data in a short time without any interruption. The maximum record length (MRL) is hardcoded in the phase and magnitude query procedure to be less than 3.700.000 samples. This value is determined after trying to read different numbers of samples that are extracted from the ASA and analyzed in a relatively short time. Hence, the criteria for identifying such a number is the timing and the correctness of queried results.

$$MNOP = \text{integer}\left(\frac{MRRL}{CRL}\right)$$

Equation (13): Calculate the maximum readable number of periods.

$$FRL = IRL \times NOP$$

Equation (14): Final record length calculation

Where:

MRRL: Maximum Readable Record Length

CRL: Corrected Record Length

MNOP: Maximum Number Of Periods

IRL: Initial Record Length

FRL: Final Record Length

The sample rate is designated according to the measurement bandwidth specified in the MCM and the record length is determined from the carrier spacing and sample rate in the MCM. The sample rate and record length must be defined for the IQ Analyser and the measurement time (Meas Time) will be calculated automatically in the IQ analyzer according to Equation (1). For more information, please check

The following TCL language command is used to apply the sample rate and record length on the IQ Analyser. This TCL command reads I/Q-values with an amount of equal to \$recordLength which starts from the trigger point equal 0 (<Pretrigger Samples> was 0). It sets the filter type to normal and the sample rate to \$sampleRate. The trigger is fed from an external instrument, the slope is positive and pretrigger samples is 0.

```
ASA1 write_read "TRAC:IQ:SET NORM,0,$sampleRate MHz,EXT,POS,0,$recordLength; *OPC?"
```

2.2.11 Query Trigger Offset

Since the R&S FSW samples the measurements internally at a much higher rate than the application, the trigger point determined internally is much more precise than the one determined from the downsampled data in the application. The **Trigger Point In Sample (TPIS)** command queries the time offset between the sample start and the trigger event.

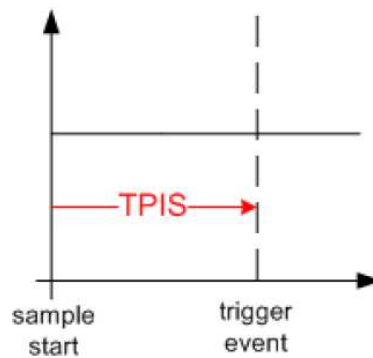


Figure 7: The difference between the sample start and trigger event.

The TPIS value can only be determined in triggered measurements using an external or IF Power trigger, otherwise, the returned TPIS value is 0.

The TPIS command is not available if the Digital Baseband Interface (in the R&S FSW-B17 option) is active and the measurement bandwidth is larger than 80 MHz. This means that the phase of any signal is queried correctly from the I/Q analyzer only and only if the analysis bandwidth is less than 80 MHz.

For an absolute group delay measurement, the TPIS command is used for calculating the offset between the sample start and the actual trigger event. This command is valid for bandwidths larger than 80 MHz. If it is intended to capture the phase of the signal correctly then the maximum bandwidth in data acquisition of the IQ analyzer is set to 80 MHz (see Figure 4). The group delay measurement is executed either with a 280 MHz spans for a *relative* group delay measurement or with a 64 MHz spans for *absolute* group delay measurement. The following command sets the appropriate IQ analyzer bandwidth. It checks whether the span is smaller than 64 MHz and subsequently sets the maximum bandwidth in the data acquisition of the IQ analyzer to 80 MHz. It also queries the trigger offset for accurate phase measurement. Note that the query trigger offset makes no sense if the span is larger than 80 MHz[7].

```
if {$options(span) < 64} {ASA1 write "TRAC:IQ:WBAN:MBW 80MHZ"} else {ASA1 write "TRAC:IQ:WBAN:MBW MAX"}

#if {$options(-span) < 64} {

    set triggerOffset [ASA1 write_read "TRAC:IQ:TPIS?"]

}
```

2.2.12 Reading IQ data from spectrum analyzer:

The next task is reading I/Q data from the analyzer. Firstly, the following command specifies the I/Q binary data format as a list of alternating in-phase and quadrature-phase samples, i.e. $I_1, Q_1, I_2, Q_2, I_3, Q_3, \dots, I_n, Q_n$.

```
ASA1 write "TRAC:IQ:DATA:FORM IQP"
```

We specify the I and Q data as 32-bit floating-point numbers. In the spectrum application, the format setting REAL is used for the binary transmission of the trace data. For I/Q data, 8 bytes per I/Q-sample are returned for this format setting.

```
ASA1 write "FORM REAL,32"
```

The structure of the binary scan TCL command is:

```
binary scan string formatString ?varName varName ...?
```

The binary scan command parses fields from a binary string and returns the number of conversions performed. *string* noted as the input bytes to be parsed (one byte per character and characters not representable as a byte have their high bits chopped) and *formatString* indicates how to parse it. Each *varName* gives the name of a variable; when a field is scanned from the string the result is assigned to the corresponding variable[10].

2.2.12.1 formatString f*:

The data is interpreted as count single-precision floating-point numbers in the machine's native representation. The floating-point numbers are stored in the corresponding variable as a list. If count is *, then all of the remaining bytes in the string will be scanned. If count is omitted, then one single-precision floating-point number will be scanned. The size of a floating-point number may vary across architectures, so the number of bytes that are scanned may vary. If the data does not represent a valid floating-point number, the resulting value is undefined and compiler dependent. For example, on an MS Windows system running on an Intel Pentium processor[10].

```
ASA1 write "FORM REAL,32"
set res [[ASA1 cget -connIfObj] write_read_scpi_block "TRAC:IQ:DATA:MEM?" -
read_mode sync -read_timeout 60000]
binary scan $res f* res
```

2.2.13 Averaging and Fast Fourier Transform (FFT)

The goal after reading the raw I and Q data from the analyzer is to calculate the magnitude and phase of the signal from the raw IQ data. Therefore, the Fourier transform of the IQ samples from the time domain to the frequency domain is needed.

Fourier transform is linear which means that implementing an averaging on the raw I/Q data in the time domain and next carrying out the Fourier transform (first method) is equivalent to carrying out the Fourier transform

firstly and afterward applying the averaging in the frequency domain (second method). This is true up to rounding errors in the computer arithmetic[9].

2.2.13.1 First Method

In the first method, first a temporal averaging of the I/Q samples is carried out and then the FFT. For example, if the IQ samples represent 1000 periods of a periodic signal and we have 2048 samples in a single period, we sum up the first IQ sample of each period and divide the sum by the number of periods (in this example: 1000). The same procedure is carried out for each subsequent sample of the period until the last sample of the first period. In this way, we calculate one averaged period of the signal in the time domain of 2048 samples. Afterward, the Fourier transform with FFT size 2048 is carried out on the averaged IQ samples.

2.2.13.2 Second method

With the same numbers as in the example of the First Method, we apply a large Fourier transform of size 2048000 on all I/Q samples. The FFT will collect the averaged value by decimation in every 1000th bin in the frequency domain. Figure 8, Figure 9, and Figure 10 show the periodic property of the frequency domain signal which is retrieved from the periodic time-domain signal. In these figures, the raw frequency domain amplitude of the MC chirp signal with carrier spacing of 0.1 MHz are calculated from 3,5, and 10 periods of the time domain signal.

Note: you could find the TCL code for implementing Fourier transform in the attachment.

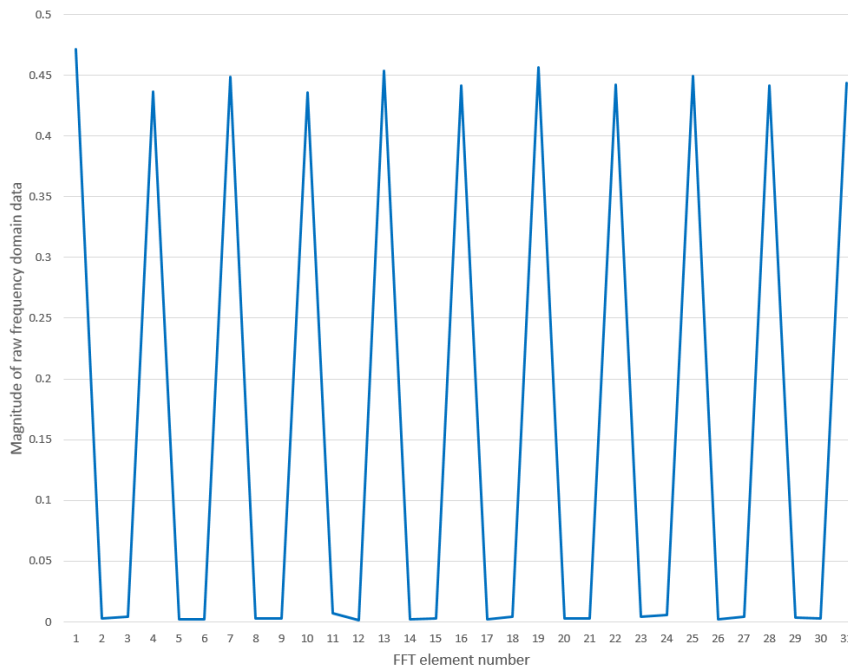


Figure 8: Raw frequency domain amplitude measured from 3 periods of the time-domain signal.

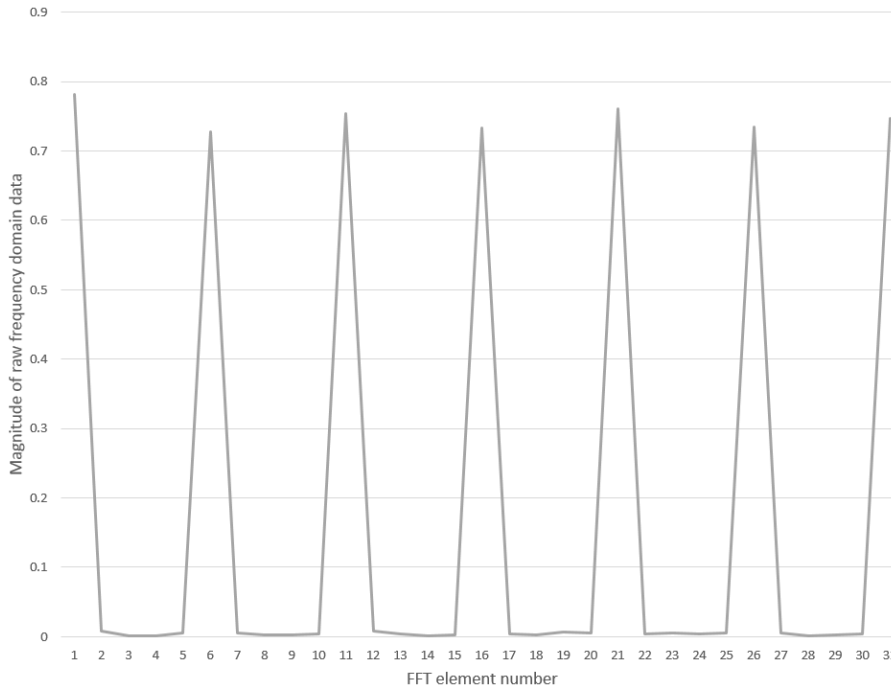


Figure 9: Raw frequency domain amplitude measured from 5 periods of the time-domain signal.

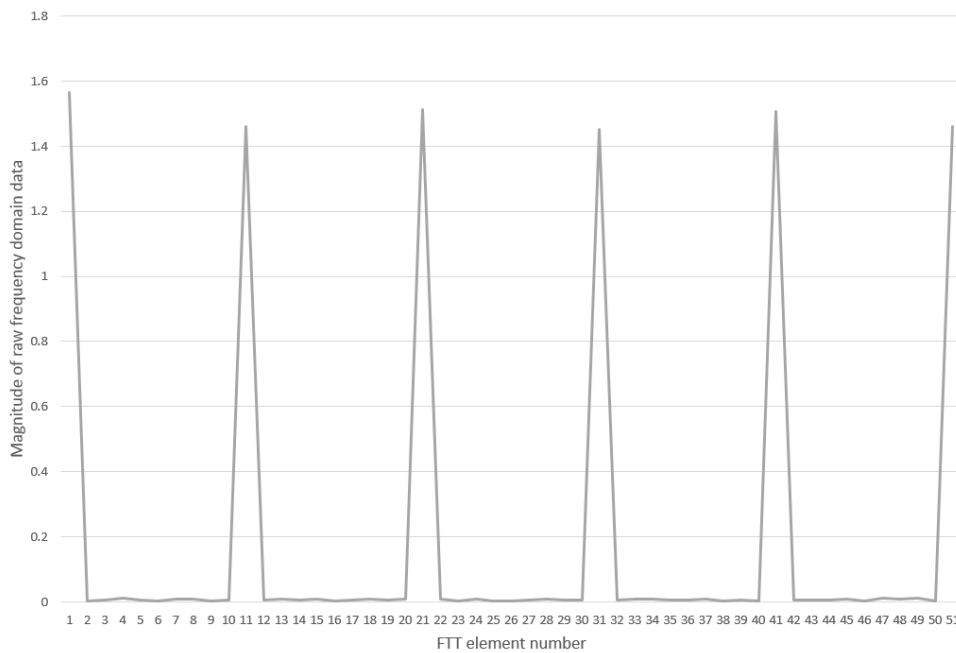


Figure 10: Raw frequency domain amplitude measured from 10 periods of the time-domain signal.

2.2.14 First and second method comparison

2.2.14.1 Result comparison

In order to prove that both methods lead to the same result, a chirp signal is set up as a stimulus with AWG at center frequency = 27725, span= 50 MHz, and spacing = 0.1MHz and the phase and magnitude of the signal versus frequency is queried in IQ Analyzer mode of the ASA with the following command:

```
ASA1.getdata_mcsignals_iq -avgcnt 1000 -span 60 -att 5 -refLevel -20 -center 27725
```

Where:

- avgcnt: Number of the displayed period on I/Q-Analyzer screen
- span: Span of the measured signal
- att: RF attenuation of the ASA
- refLevel: Reference Level of the ASA
- center: Center frequency of the ASA

Both methods are implemented with just one time running the above command. This command calls the ASA1.getdata_mcsignals_iq procedure for determining the correct sample rate and record length, query trigger offset, averaging and applying Fourier transform. With 1000 periods of the time domain IQ pairs are read from ASA. The span of the extracted data is defined to be 60 MHz. The center frequency, RF attenuation and reference level of the ASA are set to be 27725 MHz, 5 dB, and -20 dBm, respectively. Figure 12 shows the result of the measured magnitude versus frequency for both methods is identical. The phase versus frequency for the first and second method as shown in Figure 13 are identical as well.

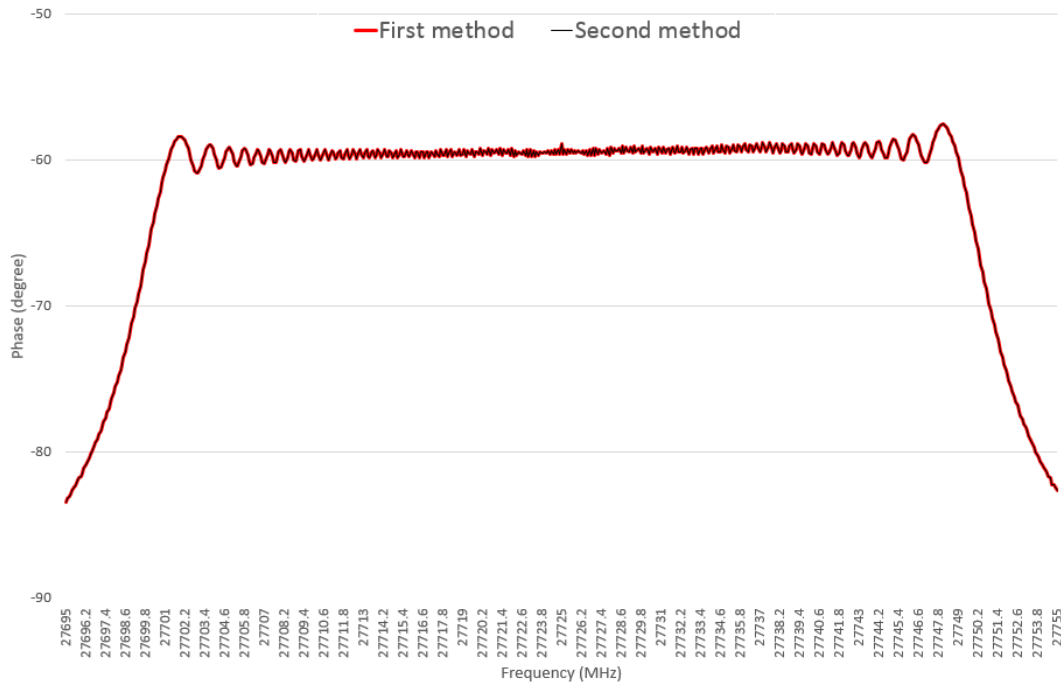


Figure 11: A comparison of calculated amplitude from IQ data for the first and second methods

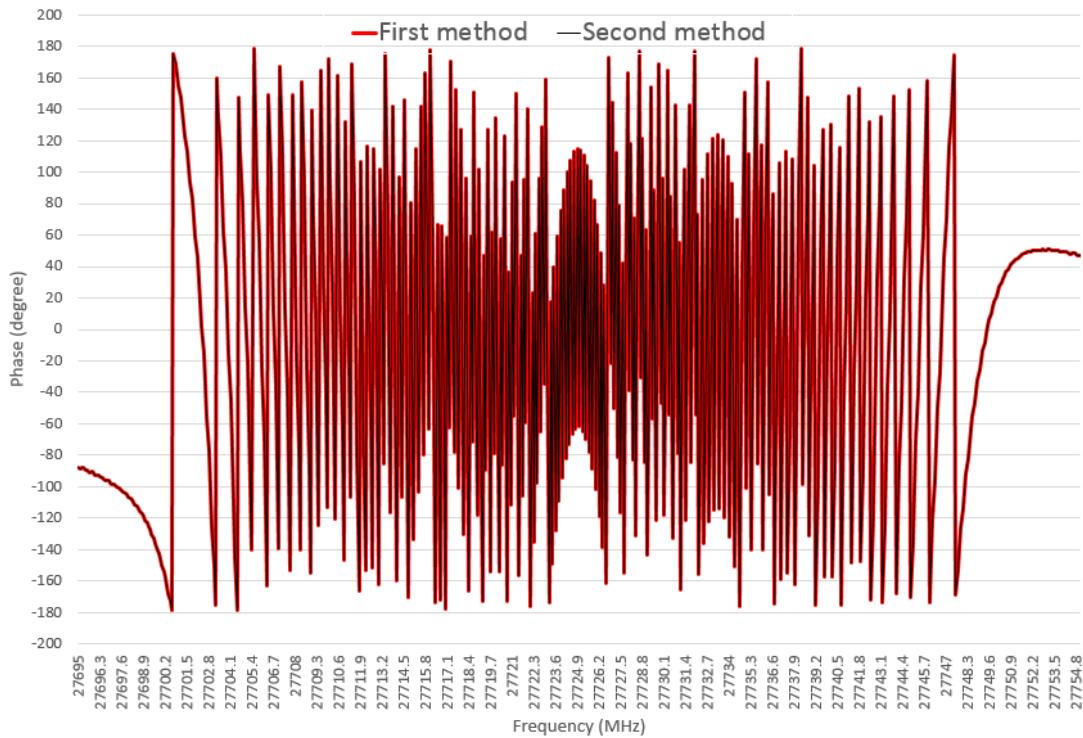


Figure 12: A comparison of the calculated phase from IQ data for the first and second methods.

2.2.14.2 Time comparison

Timing is a determining factor in many measurement procedures. Three measurements are executed sequentially with different volumes of data for processing. In each measurement, both methods are implemented separately. The duration of each method is queried from timestamps in the measurement routine. These measurements are carried out with the same analysis bandwidth, sample rate and chirp signal configuration as the stimulus signal. But they are measured with a different number of periods (1000, 5000 and 10000 numbers of periods which result in different processing data volumes). Since the results of both methods are equal, our determining factor for selecting the proper method is processing time consumption. The methods are compared in terms of timing. As shown in Figure 13, Figure 14 and Figure 15 the results for different numbers of periods confirm that the second method (first the Fourier transform then averaging) is much faster than the first method. As an example, with 10000 periods, the first method requires almost 6.7 seconds and the second method needs only 3.1 seconds.

```

15:25:20.590 H: triggerOffset=0.0000001732 seconds or 17.320ns
15:25:21.111 H: Reading and converting in Binary mode takes 0.30900001525878906
15:25:21.112 H: converting takes 0.06699991226196289
15:25:21.789 H: first method takes 0.6760001182556152 seconds
15:25:22.124 H: second method takes 0.3340001106262207 seconds

```

Figure 13: Timing of the first method and second method with 1000 periods

```

15:27:38.134 H: triggerOffset=0.0000002204 seconds or 22.040ns
15:27:39.387 H: Reading and converting in Binary mode takes 1.0440001487731934
15:27:39.388 H: converting takes 0.33100008964538574
15:27:42.478 H: first method takes 3.0889999866485596 seconds
15:27:44.027 H: second method takes 1.5479998588562012 seconds

```

Figure 14: Timing of the first and second method with 5000 periods

```

15:28:44.690 H: triggerOffset=0.0000001566 seconds or 15.660ns
15:28:47.013 H: Reading and converting in Binary mode takes 2.121999979019165
15:28:47.013 H: converting takes 0.6689999103546143
15:28:53.766 H: first method takes 6.752000093460083 seconds
15:28:56.930 H: second method takes 3.1630001068115234 seconds

```

Figure 15: Calculated phase from IQ data second method with 10000 periods

2.2.15 Magnitude and phase calculation

The Fast Fourier transform results in a complex-valued representation of the signal which is composed of real and imaginary parts, or equivalently with magnitude and phase. The magnitude of the signal representation is expressed in the pseudo unit decibels relative to 1 mW (dBm) and the phase of the signal is expressed in the unit degree. They are determined with the following equations:

$$scale\ factor = \frac{1000}{50 \times (length\ of\ fft)^2}$$

$$Magnitude(dBm) = 10 \times LOG_{10}(scale\ factor \times (Re^2 + Im^2))$$

$$Phase(degree) = atan2\left(\frac{Im}{Re}\right) \times \left(\frac{45}{atan(1)}\right)$$

Where:

- $atan2(y/x)$ returns the arc tangent of y/x , in the range $[-\pi, \pi]$ radians and $atan(y/x)$ returns the arc tangent of y/x , in the range $[-2, \pi/2]$ radians[10].
- $\left(\frac{45}{atan(1)}\right)$ is equal to π [10].

2.2.16 Magnitude verification

In the following, we prove that the magnitude of the signal is measured correctly. A simple comparison is considered by measuring the magnitude of the same chirp signal spectrum in the SAN Mode and IQ Analyzer Mode of the ASA. Firstly, a chirp signal with the configuration shown in Figure 16 is applied. The power level of the chirp signal for some carrier near the center frequency is measured in SAN Mode of ASA with a proper RBW and SPAN, see Figure 17. Secondly, the same present signal data measured with IQ Analyzer Mode of ASA. The magnitude of the chirp signal is calculated by calling the IQ Analyzer method of the measured spectrum with the following command:

```
ASA1.getdata_mcsignals_iq -avgcnt 1000 -span 169.6 -att 5 -refLevel -20 -center 14187.5
```

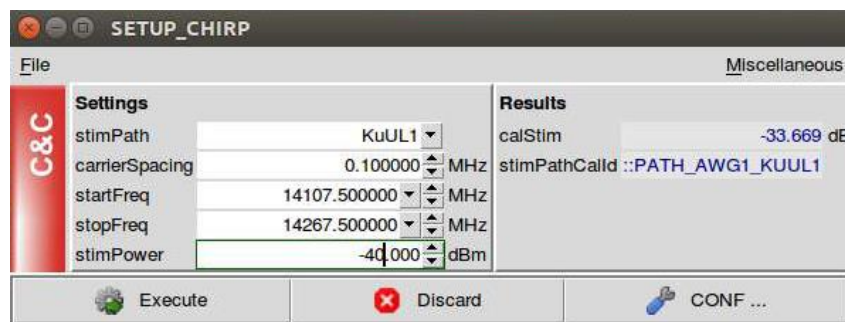


Figure 16: Configured parameter for chirp signal.

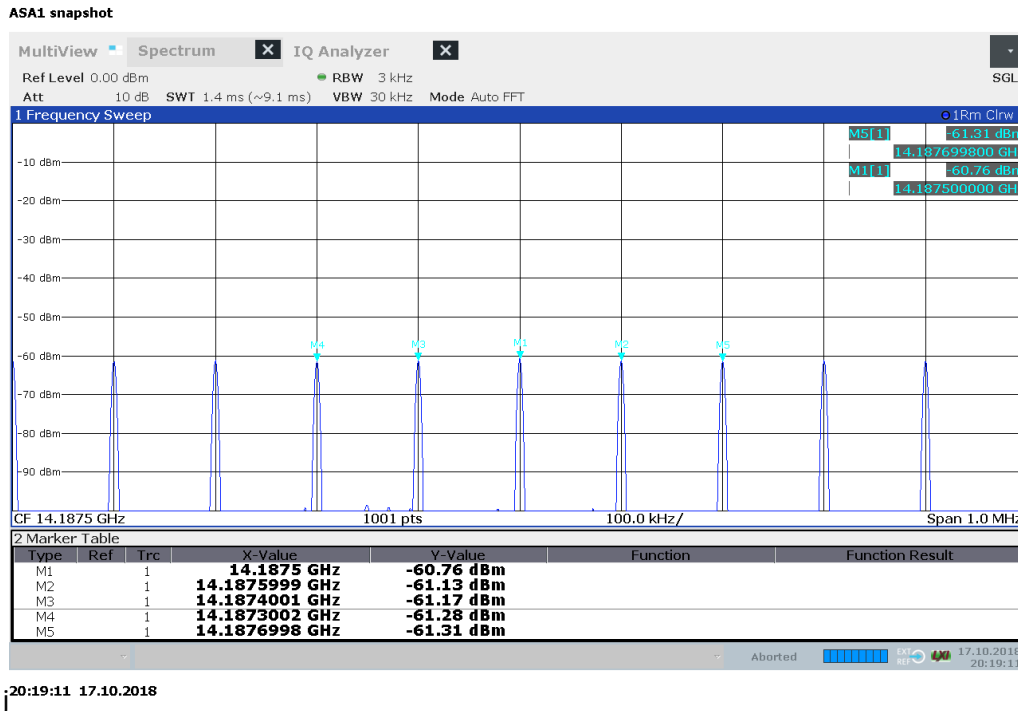


Figure 17: Measured single carrier of chirp signal in SAN modus of ASA.

The measured magnitude of the chirp signal with I/Q-Analyzer is shown in Figure 18. In Table 1, the magnitude of five carriers compared for the same chirp signal measured at ASA in two different modes through the internal loop. These comparisons show how accurate we are in measuring the magnitude of the chirp signal. In the worst case, the difference of 0.08 dB certifies the accuracy of measuring the magnitude of the spectrum in IQ Analyzer Mode.

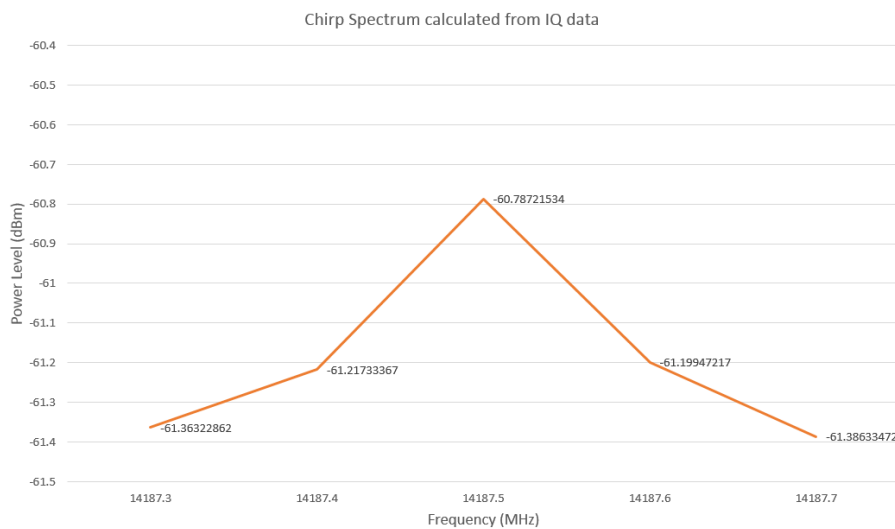


Figure 18: Measured five carriers of chirp signal in IQ Analyser mode of ASA.

Method \ Frequency (MHz)	SAN	IQ Analyzer	Difference
14187.5(center)	-60.76 dBm	-60.78 dBm	0.02 dB
14187.6	-61.13 dBm	-61.19 dBm	0.06 dB
14187.4	-61.17 dBm	-61.21 dBm	0.04 dB
14187.3	-61.28 dBm	-61.36 dBm	0.08 dB
14187.7	-61.31 dBm	-61.38 dBm	0.07 dB

Table 1: Chirp spectrum magnitude comparison, measured with IQ analyzer and SAN.

2.2.17 Phase verification

The comparison of the measured phase with the K17 Option and IQ Analyzer method is verified by an absolute group delay measurement.

First, the MCC is carried out with the same setting as the MCM (check Chapter 5 and Chapter 6 to get information about MCC and MCM, respectively). Next, the obtained MCC is used as a reference for the group delay measurement. The group delay difference resulting from the calibrated phase value and the measured phase value must be zero seconds. The difference is not exactly zero seconds due to systematic and environmental errors that instruments are prone to.

In the absolute group delay measurement, we are shifting the almost 0s group delay by our calculated group delay of the TAC for the FWD channel which has been measured with the VNA. Therefore, our criterion for verifying the phase measurement is the group delay measurement with a bandwidth of fewer than 80 MHz. The trace of the group delay versus frequency (determined in the MCM) shows the same value and does not change dramatically from one measurement to another.

For verifying the phase measurement, the MCC is executed once with a bandwidth of 56 MHz (one out of five sub spans). Afterward, three MCGD measurements are executed consecutively, as shown in Figure 19. However, the final group delay is shifted by a constant measured group delay value of DUT. Each measurement results in different group delay versus frequency with the same DUT. This shows leakage in our phase measurements.

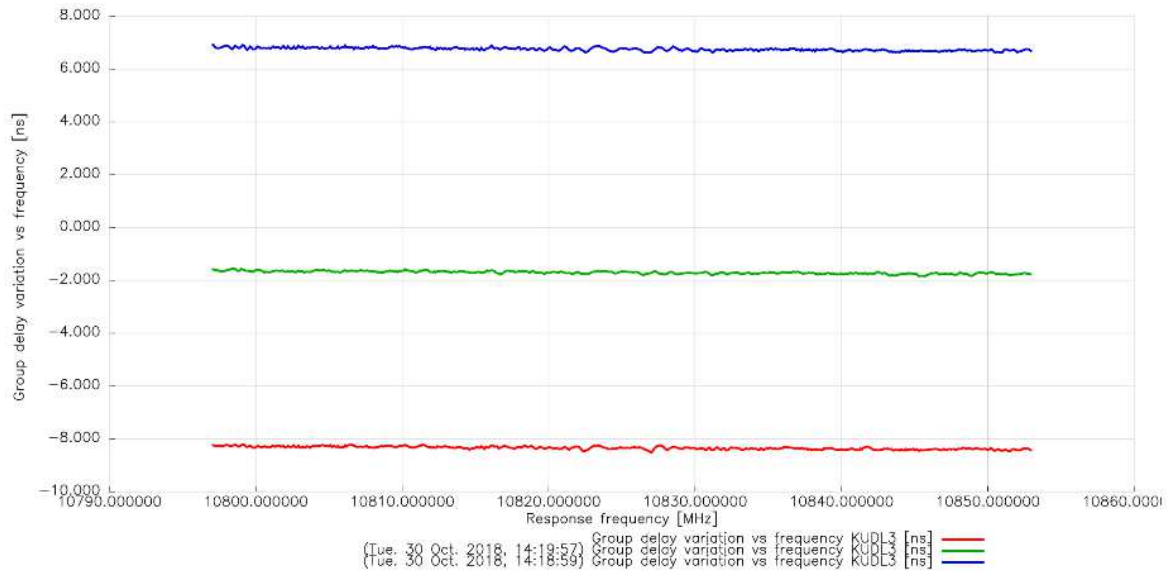


Figure 19: Three consecutive group delay measurements before phase correction.

During an in-depth investigation, we observed that there are trigger offsets between the start of our sample and the actual trigger event. Therefore, we need to inquire about the trigger offset from ASA and apply the inquired trigger offset in order to correctly calculate the group delay from the measured phase of the signal. Please check §2.2.18 for more information.

Below, we discuss how the inquired trigger offset is correctly applied to the group delay measurements.

The inquired trigger offset is specified in the time domain in seconds. The obvious method for applying the inquired trigger offset to our group delay measurement is to calculate the difference between the group delay in the time domain and the trigger offset. The problem of such a calculation is that the trigger offset is always considered as an offset in our group delay measurement, not in the original phase measurement. The phase of the signal is always measured wrongly. Hence, it is much better to implement the trigger offset correction in the phase measurement.

Group delay is defined as the negative slope of the phase curve versus frequency, hence, the correct method to apply the inquired trigger offset in our phase measurement is to differentiate it from the slope of the phase versus frequency. This results in correct phase measurement and consequently correct group delay measurement. (See Figure 20)

2.2.18 Slope correction

Group delay is a measure of phase distortion. It is defined as the negative derivative of the phase curve versus frequency:

$$\tau(t) = - \frac{d\varphi(\text{Radian})}{d\omega\left(\frac{\text{Radian}}{\text{Second}}\right)} = - \frac{\frac{2\pi}{360} \times d\varphi(\text{Degree})}{2\pi \times df(\text{Hz})} = - \frac{d\varphi(\text{Degree})}{360 \times df(\text{Hz})}$$

Equation (15): Group delay is driven from phase and frequency

$$\text{Trigger offset} = - \frac{\text{slope corrector}}{360 \times \text{spacing of chirp signal}(\text{frequency steps})}$$

Equation (16): Trigger offset and slope corrector relation.

$$\text{Slope corrector} = -360 \times \text{spacing of chirp signal} \times \text{trigger offset}$$

Equation (17): Slope corrector factor is driven from trigger offset.

Where:

$\tau(t)$ = group delay or trigger offset

$d\varphi$ = phase shift

$d\omega$ = frequency

The slope correction is calculated according to the Equation (17), Subsequently, the slope corrector has to be differentiated from the slope of our phase measurement.

After implementing the proposed phase slope correction, the same group delay measurement as shown in Figure 19 with a bandwidth of 56 MHz is executed three times consecutively. The results for the group delay as shown in Figure 20, are very close to each other (with a tolerance of approx. 100ps) which validates that the phase is measured accurately.

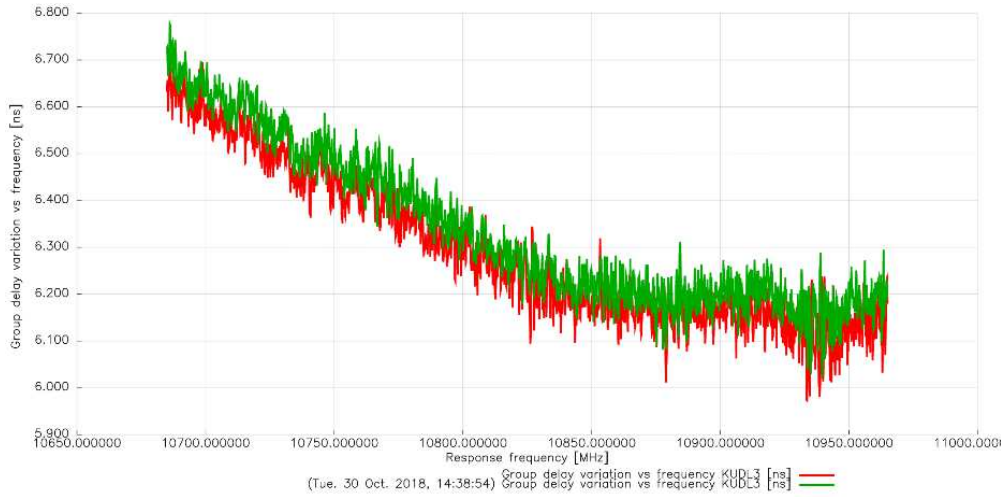


Figure 20: Three consecutive group delay measurements after phase correction.

In the following, we have measured the magnitude and phase of the chirp signal with the K17 option and IQ Analyzer, see Figure 21, Figure 22, Figure 23 and Figure 24.

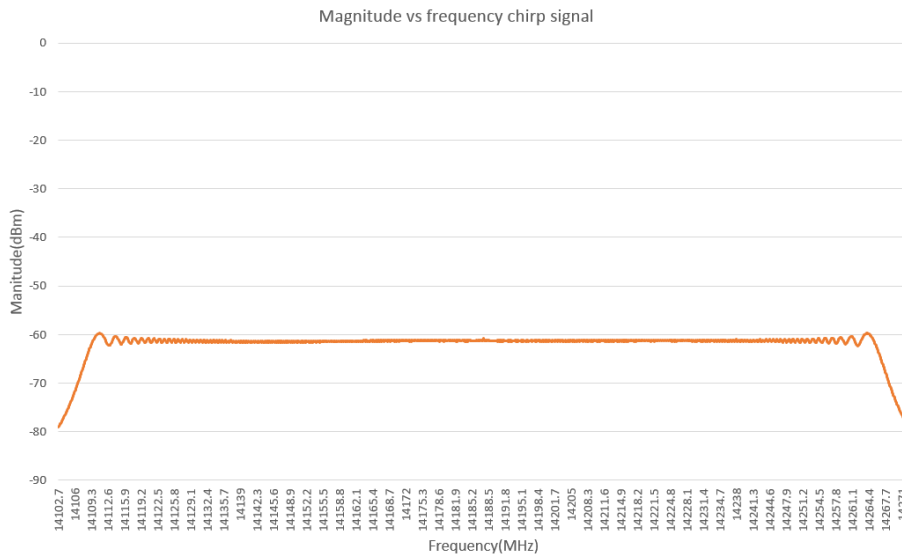


Figure 21: Magnitude of the chirp signal measured with I/Q-Analyzer measurement.

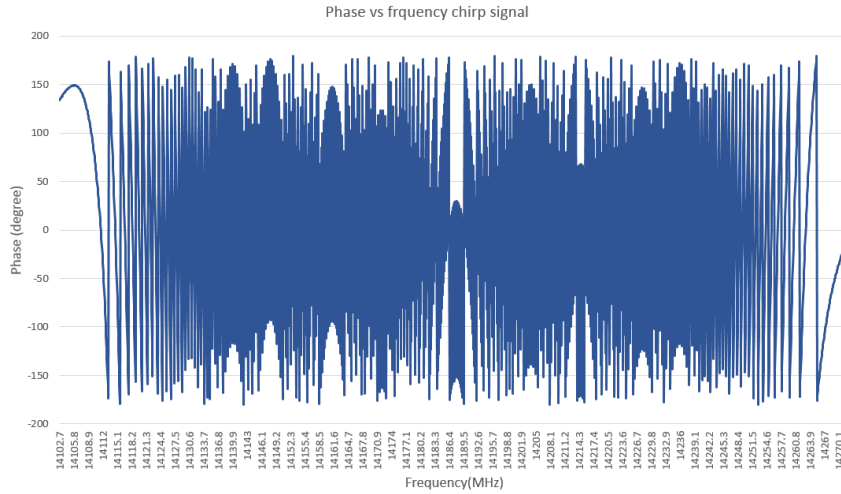


Figure 22: Phase of the chirp signal measured with I/Q-Analyzer.

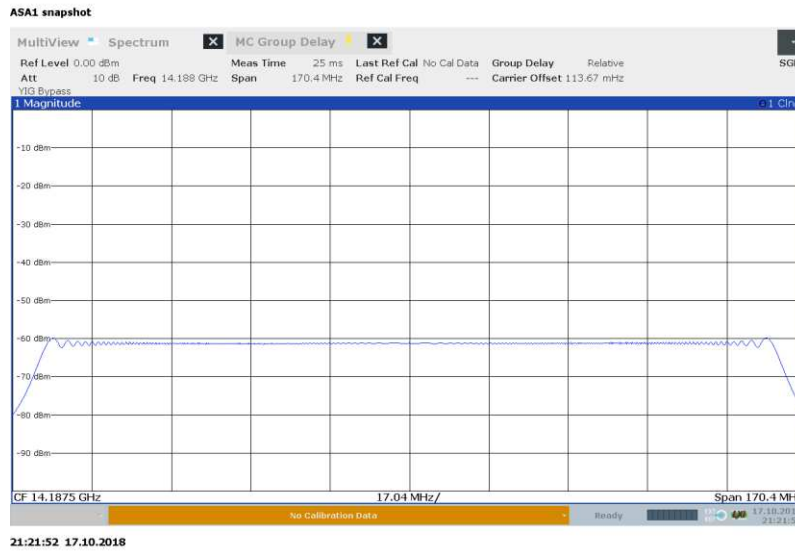


Figure 23: Magnitude of the chirp signal measured with the K17 option.

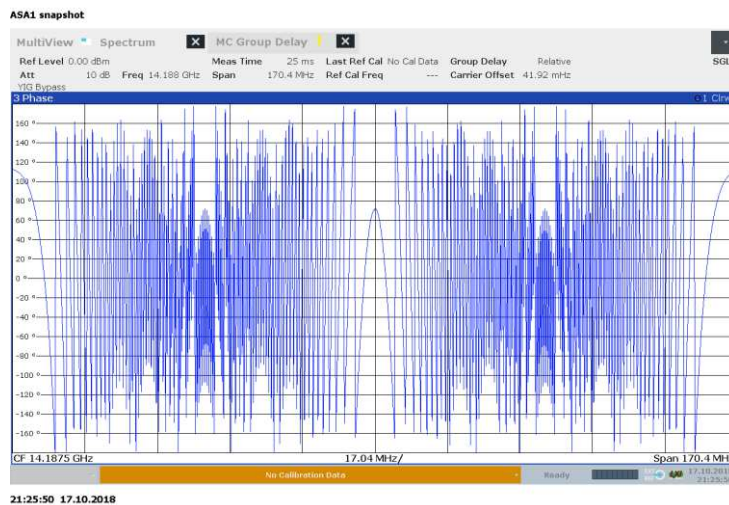


Figure 24: Phase of the chirp signal measured with the K17 option.

2.3 K17 option and IQ-Analyzer comparison

2.3.1 Different spacing

The phase and magnitude of the signal have already been verified, the magnitude and phase of the chirp signal are checked for different bandwidths of the waveform. However, with the same spacing of the carrier, the question that is still open is whether the implemented method to measure the phase and magnitude of the chirp waveform is capable to retrieve data correctly for a particular spacing of the carrier?" to answer this question, we will apply chirp signal for all variant of carrier spacing, execute and compare the MC gain and group delay measurement with K17 option and with IQ Analyser measurement.

For all variants of possible spacing, the calibrations and measurements are executed with the K17 option and I/Q-Analyser. In Figure 25, Figure 26, Figure 27, Figure 28, Figure 29, Figure 30, Figure 31, Figure 32, Figure 33, and Figure 34, the green curves represent the K17 option measurements and red curves stand for I/Q-Analyser measurement. As can be seen, in MCGD measurement both curves follow the same trend through changing carrier spacing of the chirp waveform. Executing one measurement after another with the same measurement setup and routine will result in very similar absolute group delay traces. However, the transmission gain traces are slightly different. There are so many parameters that play a role in the gain measurement accuracy. In each setup of the MC chirp signal, there is a possibility of the non-identical AWG output power level. The temperature, location, and position of the cables are also determining factors in the measured value of the transmission gain.

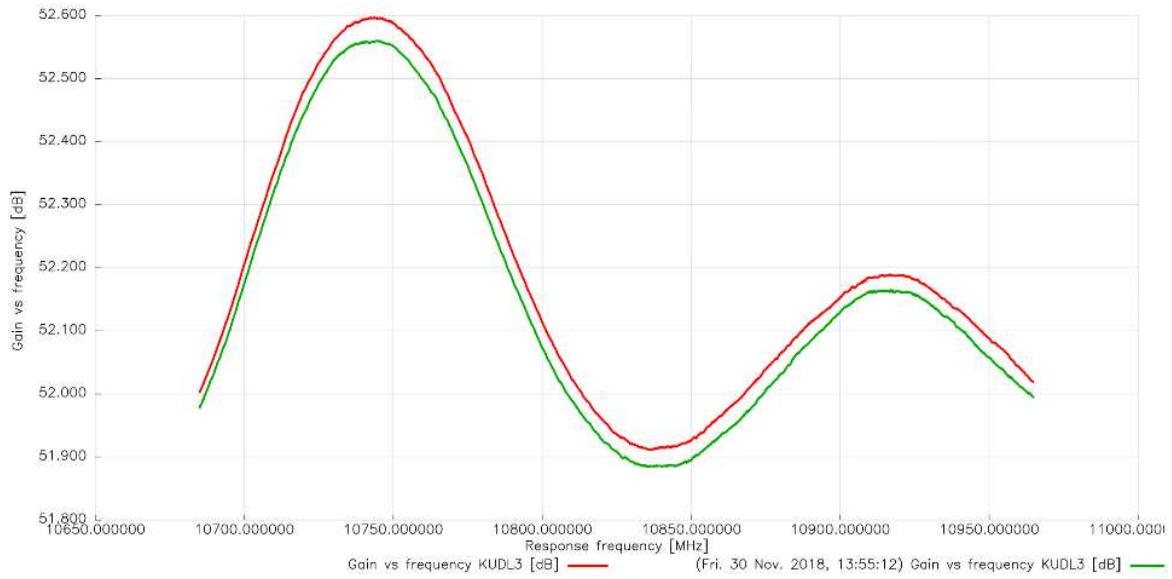


Figure 25: Transmission gain measurements with carrier spacing of 0.1 MHz (red curve with I/Q-Analyser amplitude acquisition, green curve with K17 amplitude acquisition).

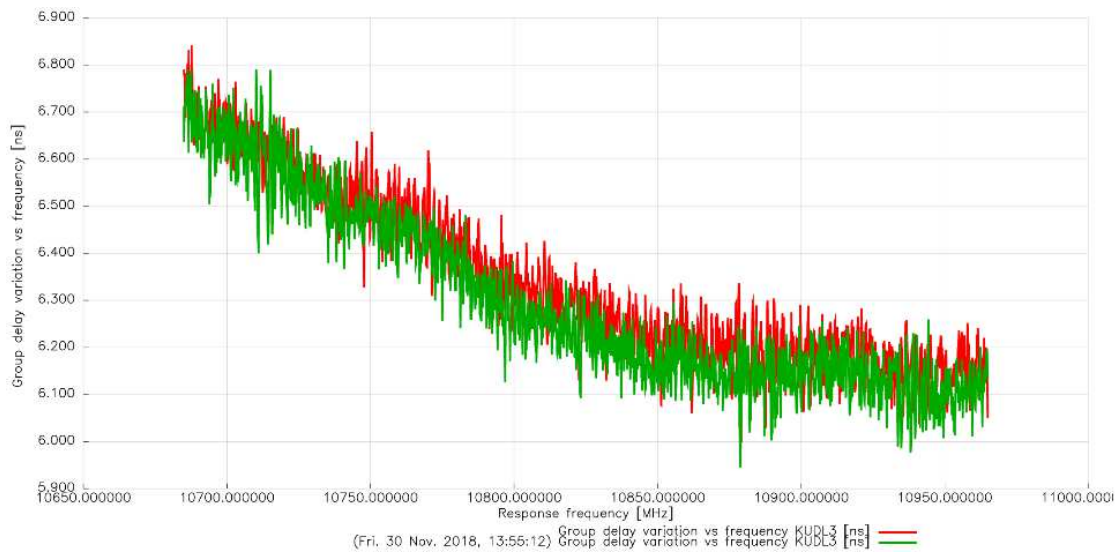


Figure 26: Absolute group delay measurements with carrier spacing of 0.1 MHz (red curve with I/Q-Analyser phase acquisition, green curve with K17 phase acquisition).

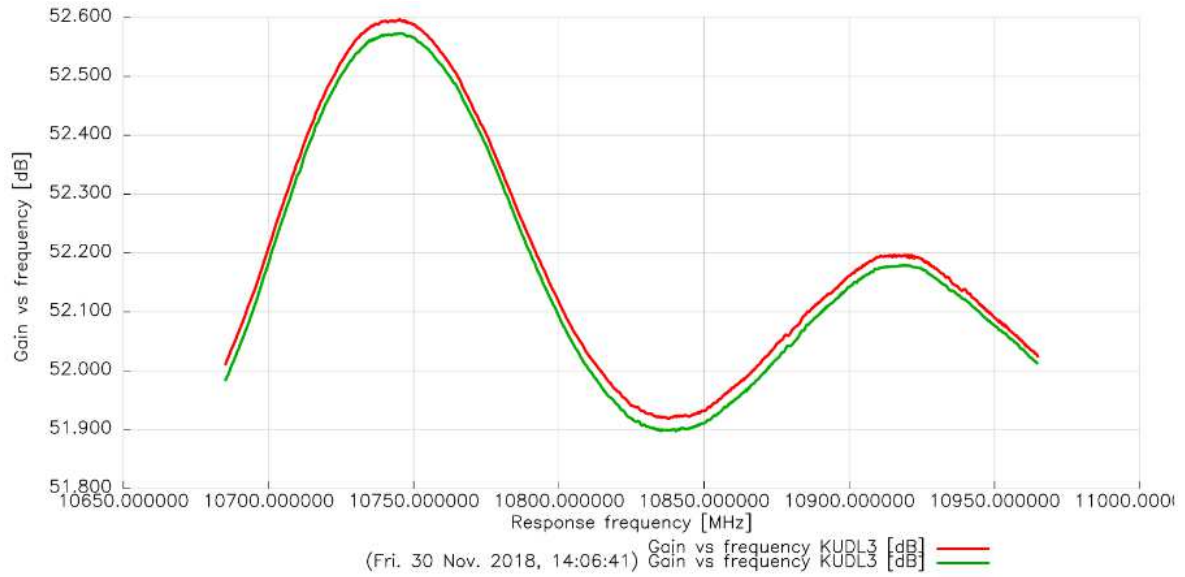


Figure 27: Transmission gain measurements with carrier spacing of 0.2 MHz (red curve with I/Q-Analyser amplitude acquisition, green curve with K17 amplitude acquisition).

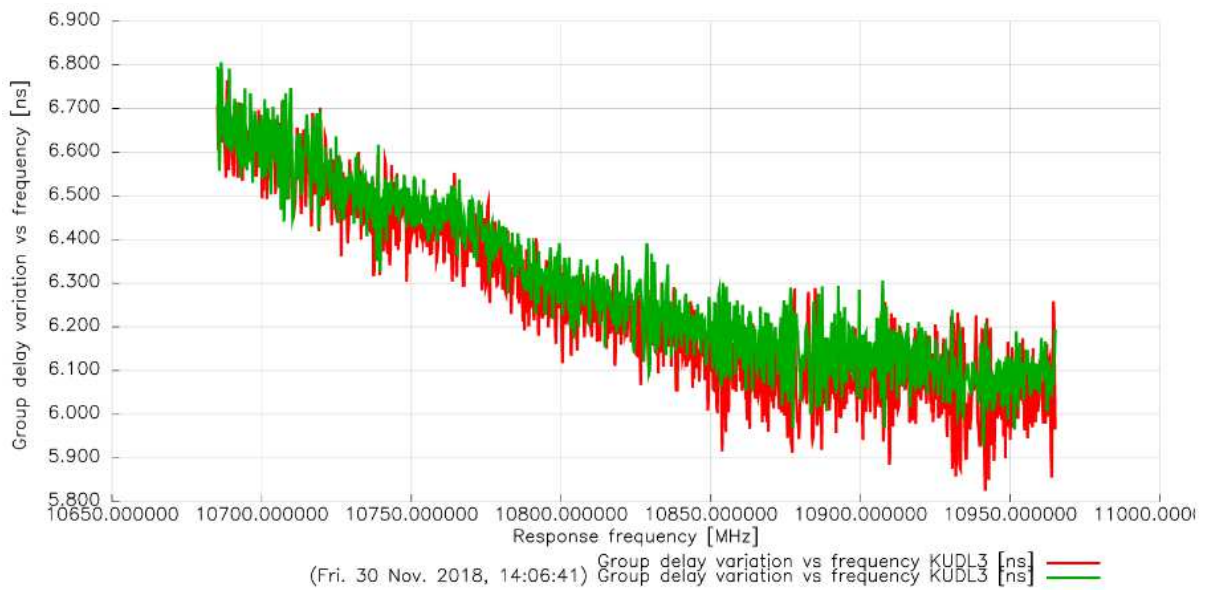


Figure 28: Absolute group delay measurements with carrier spacing of 0.2 MHz (red curve with I/Q-Analyser phase acquisition, green curve with K17 phase acquisition).

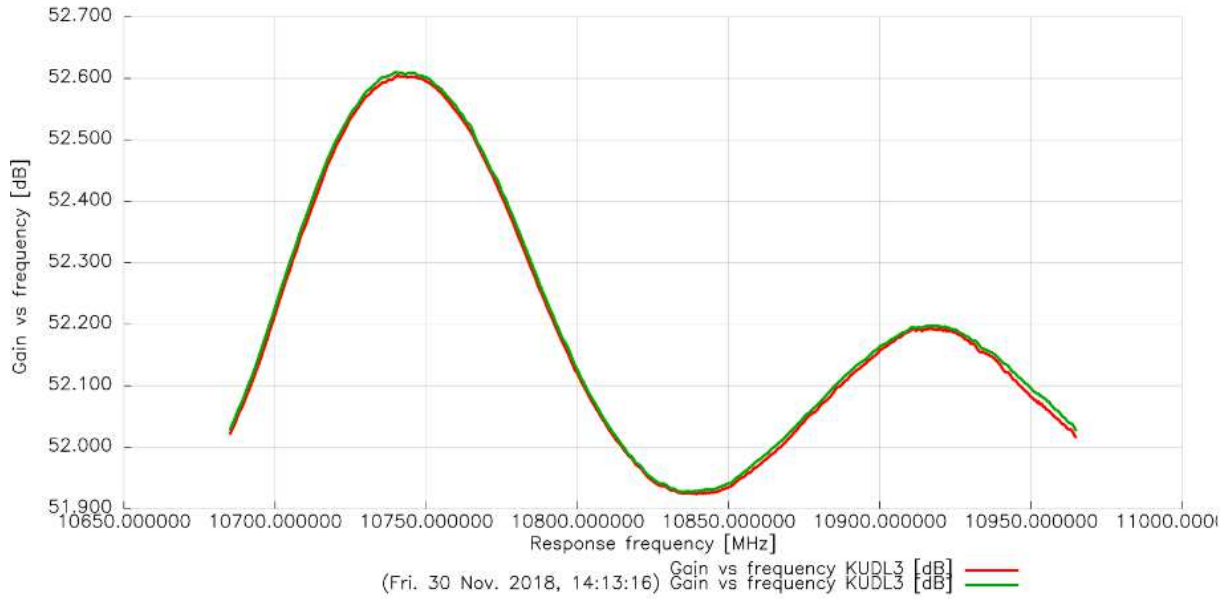


Figure 29: Transmission gain measurements with carrier spacing of 0.5 MHz (red curve with I/Q-Analyser amplitude acquisition, green curve with K17 amplitude acquisition).

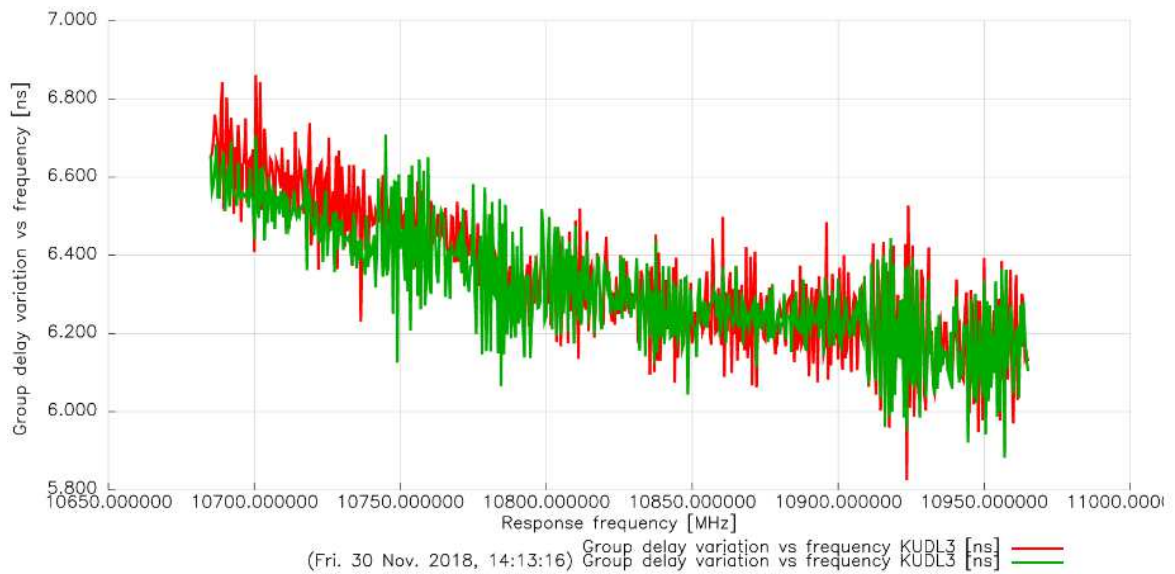


Figure 30: Absolute group delay measurements with carrier spacing of 0.5 MHz (red curve with I/Q-Analyser phase acquisition, green curve with K17 phase acquisition).

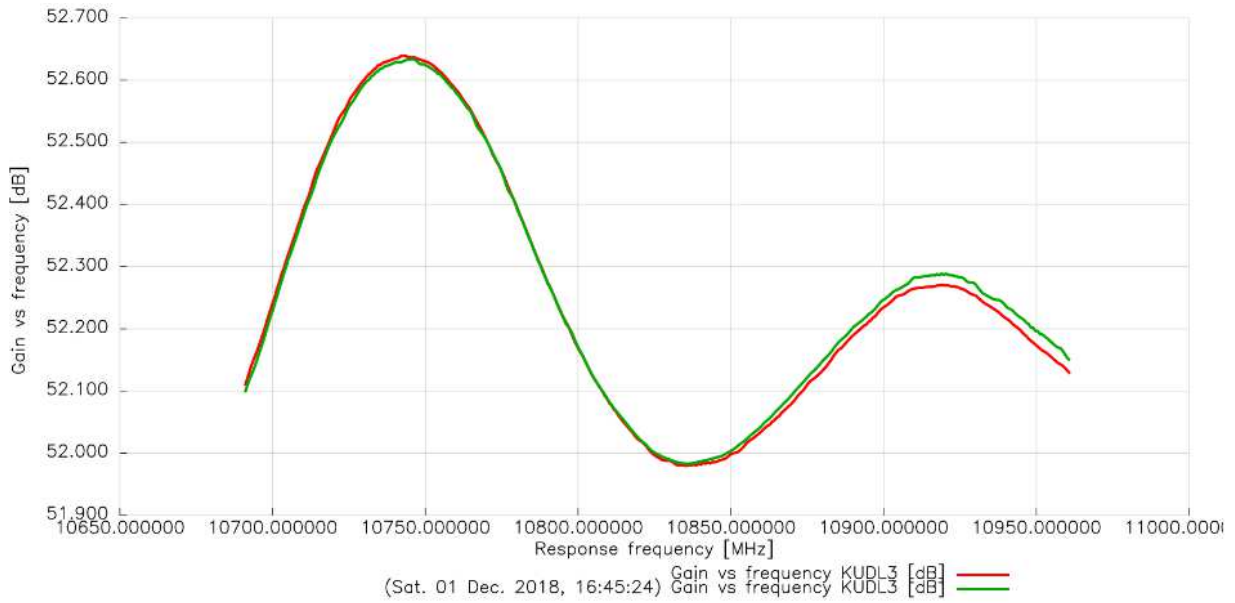


Figure 31: Transmission gain measurements with carrier spacing of 0.75 MHz (red curve with I/Q-Analyser amplitude acquisition, green curve with K17 amplitude acquisition).

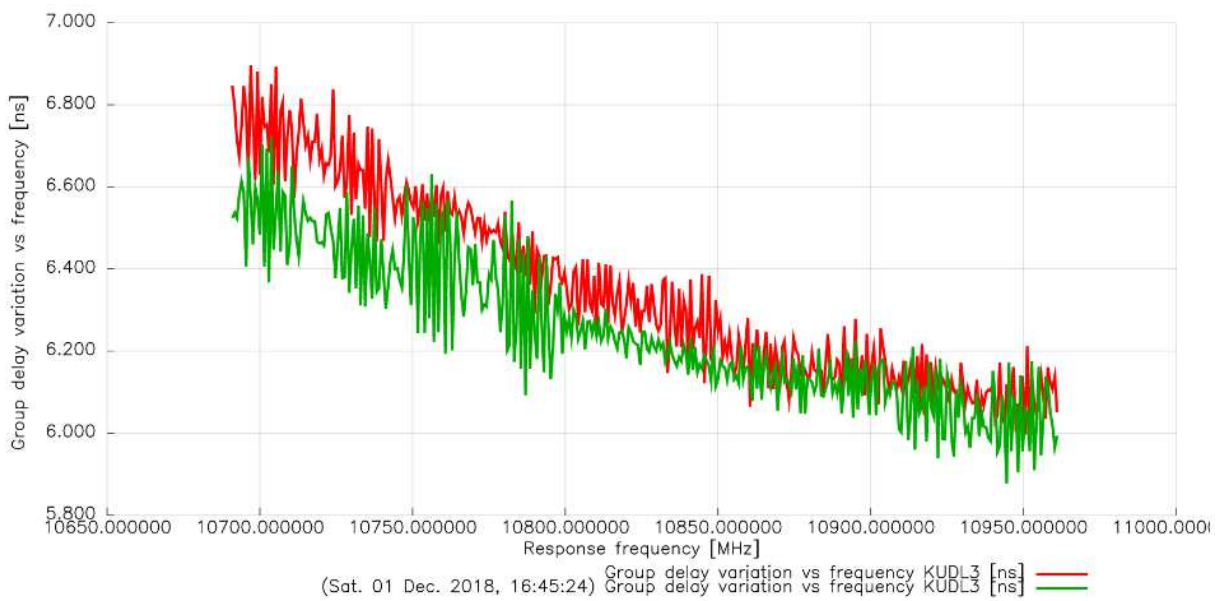


Figure 32: Absolute group delay measurements with carrier spacing of 0.75 MHz (red curve with I/Q-Analyser phase acquisition, green curve with K17 phase acquisition).

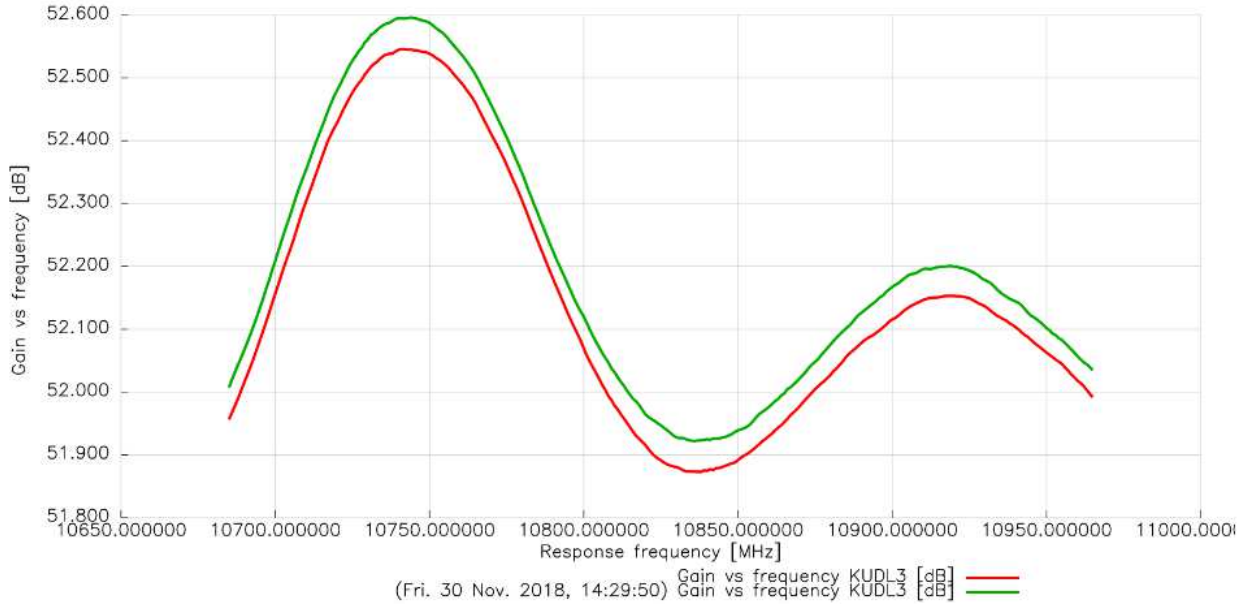


Figure 33: Transmission gain measurements with carrier spacing of 0.75 MHz (red curve with I/Q-Analyser amplitude acquisition, green curve with K17 amplitude acquisition).

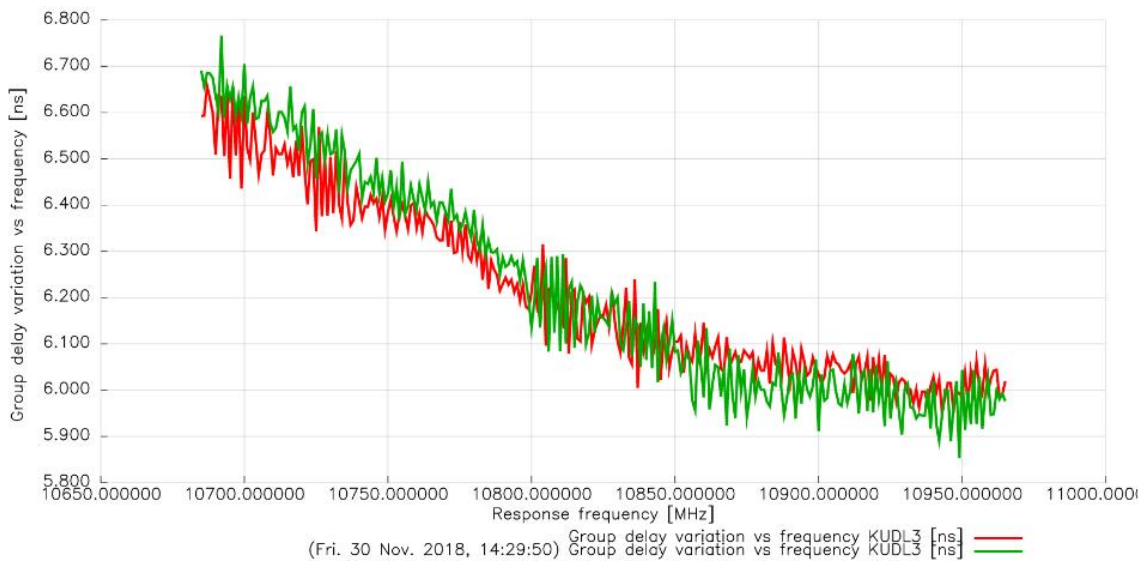


Figure 34: Absolute group delay measurements with carrier spacing of 1 MHz (red curve with I/Q-Analyser phase acquisition, green curve with K17 phase acquisition).

Table 2 shows a comparison of the ripple in absolute group delay measurements for the K17 option and I/Q-Analyser.

Method Spacing (MHz)	Group delay ripple K17 measurements	Group delay ripple IQ-Analyzer measurements	Difference
0.1	0.39 ns	0.33 ns	-0.06 ns
0.2	0.33 ns	0.40 ns	0.07 ns
0.5	0.58 ns	0.35 ns	-0.22 ns
0.75	0.48 ns	0.32 ns	-0.16 ns
1	0.21 ns	0.24 ns	0.03 ns

Table 2: IQ-Analyzer measurement and K17 option comparison for different spacing in measuring group delay.

Chapter 3

Measurement prerequisites

3.1 Payload test system overview

The RF SCOE contains the following equipment:

- RF stimulus and measurement systems.
- RF payload / TTC interface cables between rack/switching unit and payload ports.
- Measurement science description of test types and associated scripts and databases.

The OneWeb payload is basically a bent-pipe type payload. It is composed of two main links, A ForWarD(FWD) link, and a ReTurN (RTN) link.

The Forward link (FWD) is composed of:

- 2 uplink Ka-Band Gateway steerable antennas (LHCP and RHCP) and LNA receiving 8 Ka-Band 280 MHz channels each.
- 28-channels demux and flexible channelizer to translate each of the 8 Ka-band channels to 8 Ku-Band channels.
- 16 Ku band 10W SSPA operating in ALC mode.
- A 16 beams Ku Band Venetian Blind transmit user antenna

The Return link (RTN) is composed of:

- A 16 beams Ku Band Venetian Blind receive user antenna and 16 Ku Band LNA, receiving a 125 MHz channel per beam
- 28-channels multiplexer and frequency translation to translate each Ku band beam to 2x8 Ka-Band channels.
- 4 Ka-Band 3W SSPA operating in FGM mode.
- 2 downlink Ka-Band Gateway steerable antennas (LHCP and RHCP).

3.2 Multicarrier waveform

The MCGD measurement has to be carried out with a multicarrier signal. There is a vast variety of multicarrier waveforms that can be generated by an AWG to meet our demand. The criteria for choosing the best multicarrier signal actually depends on the behavior of DUT which is fed by multicarrier waveform. By taking into consideration that satellite composed of amplifiers in the RF path. These amplifiers have a linear region of constant gain e.g. G_0 where the gain is independent of input power level (small-signal gain). As the input power is increased to a level that causes the amplifier goes to the region of gain compression, the gain decreases. As every RF tester always is intended to exploit maximum gain of an amplifier.

Therefore, any deviations in the input signal over the bandwidth of interest will bring the amplifier into compression. In order to avoid such an issue, the amplitude-frequency response of the input signal of an amplifier needs to be maintained as flat as possible. Intermodulation due to non-linearity of the signal path needs to be minimized.

In the usual multicarrier stimulus, an equally spaced multi-tone stimulus with **equal amplitudes** is generated. To keep the crest factor small, the phases of the single carriers should be proportional to the square of the frequency (so-called Newman phases). The waveform has an advantage that the spectrum looks rectangular with sharp edges. The disadvantage is that the amplitude variation over time variates more than 20 dB as shown in Figure 35.

On the other hand, the chirp waveform has an equally spaced multi-tone signal with **unequal amplitudes**. The chirp waveform has the advantage that it is not sensible to non-linearity of the signal path. At a time only one CW signal is present and later the next carrier comes in the time domain which means there is no second tone to produce intermodulation. The spectrum of the chirp signal has a broader spectrum in comparison to a multi-tone signal (drawback). The amplitude over time is constant and the spectrum of the signal is not constant like a multi-tone spectrum as displayed in Figure 36. Hence, the chirp waveform is a proper candidate for utilizing it as MC stimulus in the group delay- and gain measurement of the satellite.

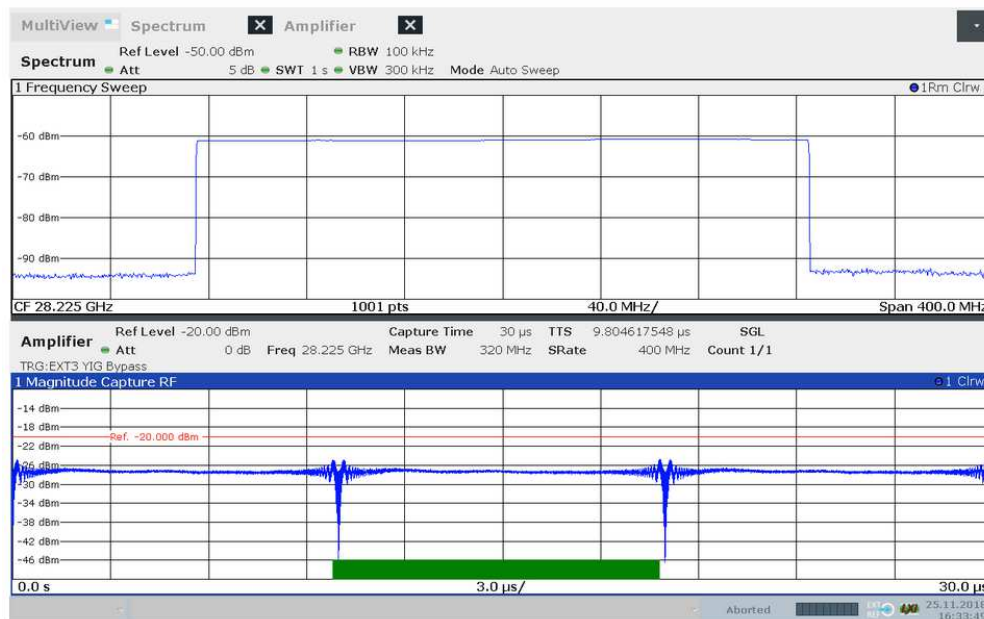


Figure 35: The spectrum and the amplitude-frequency response of the multicarrier signal.

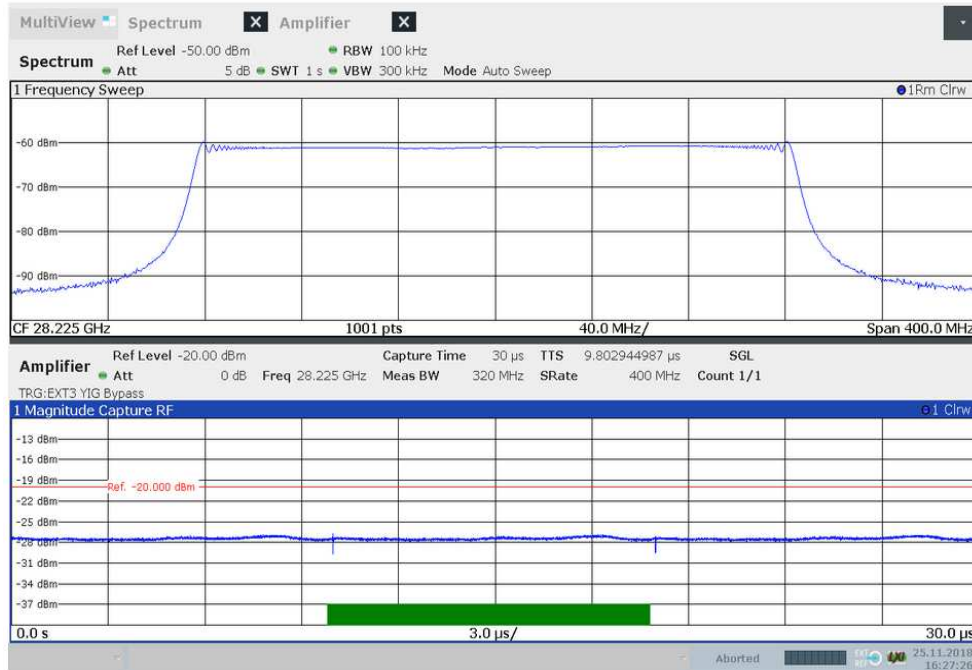


Figure 36: The spectrum and the amplitude-frequency response of the chirp signal.

Note that all measurements executed in this thesis are using 4 uplink interfaces (KaUL1, KaUL2, KaUL3, and KaUL4) and 4 downlink interfaces (KuUL1, KuUL2, KuUL3, and KuUL4). All measurements performed in the room temperature (25°C).

3.2.1 Measure transmission gain and group delay with Multicarrier stimulus

As explained previously, we planned to measure the group delay and gain of the DUT with frequency conversion with the MCGD measurement method. In the group delay and gain measurement, the output of DUT must be compared to the input signal. In MCGD measurement the output signal of the DUT is measured with ASA. There are 3 different scenarios that can be considered in this sense:

- 1- Comparing the measured output signal with a known input signal
- 2- Comparing the measured output signal with a measured input signal, this method is suggested to be used in MCGD measurement option K17.

Cons:

- Firstly, since ASA characteristic differs in measurement at different frequencies this cause an error in the measurement.
 - secondly, in OneWeb project the measurement cable changes from one measurement to another one. The signal path in the switch matrix also changes which passes through different switches, cables, and adaptors.
- 3- Calibration of a known DUT (TAC) and compare the result of calibration with measurement of the satellite relatively.

Pros:

- The same switch position or measurement setup is used in the calibration with TAC and measurement of the satellite.
- This method is very fast because the input signal measurement is removed and precise in comparison to other methods.

In Figure 37, the measurement and calibration procedures are sketched as applied for characterizing the gain and group delay responses of a DUT with frequency conversion.

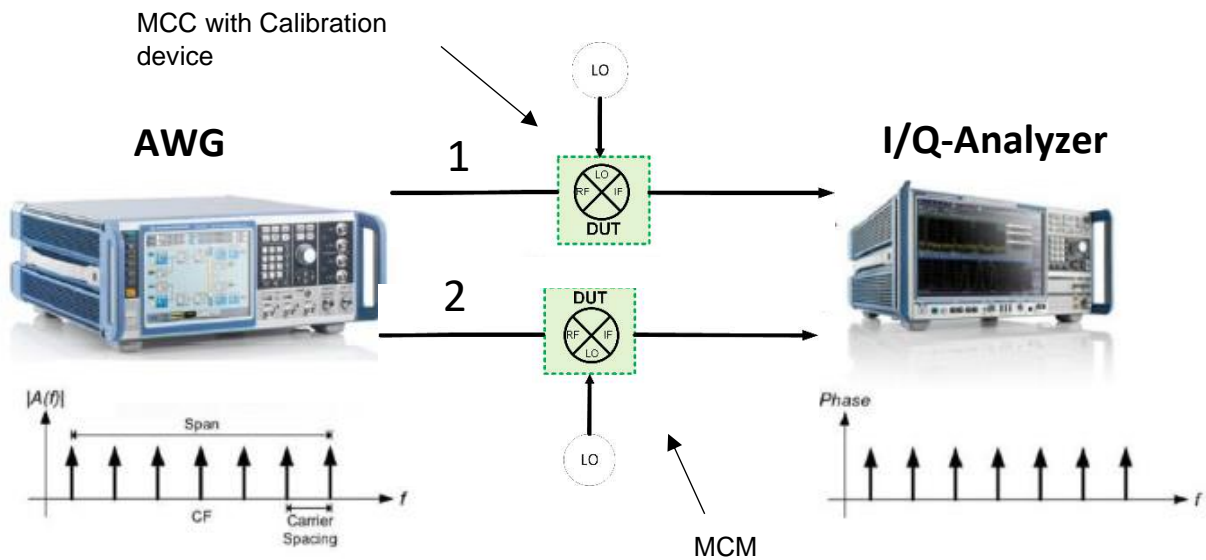


Figure 37: MCGD calibration and measurement of DUT with frequency conversion[1].

As can be seen in Figure 37, the MCC and measurements are carried out in the presence of the DUT with identical signal path configuration.

Chapter 4

Turn Around Converter Calibrations

The TAC calibrations are used as a calibration reference for the MCMs. The routine CAL_TAC calibrates the amplitude- and phase characteristics of the TAC and stores them in the database. An additional network analyzer will be needed for executing the phase calibration. The schematic of the TAC is depicted in Figure 38.

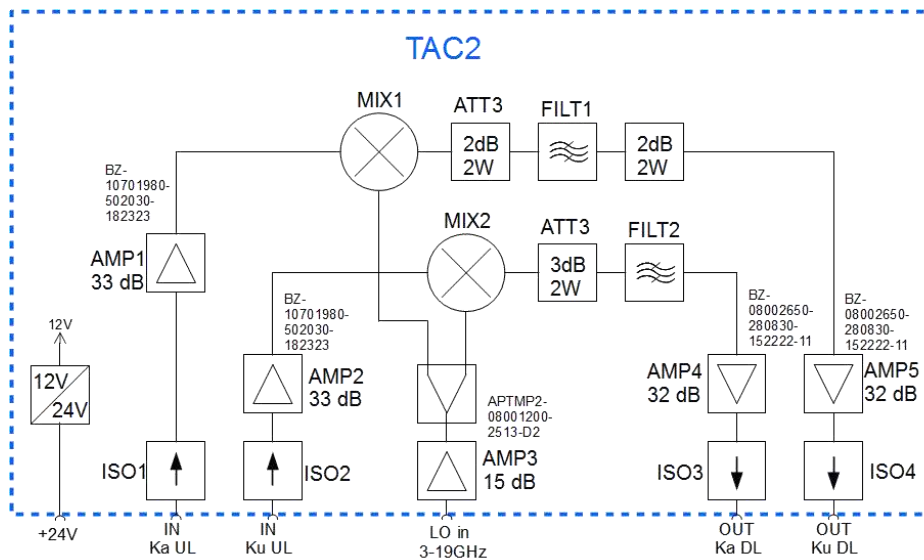


Figure 38: Schematic of The TAC[11].

4.1.1 Gain calibration

The gain responses are measured with PWS1 and PWS2 at all possible LO frequencies for all uplink- and downlink channel combinations.



Figure 39: TAC gain calibration configuration.

The gain calibration of the TAC uses two power sensors. Both power sensors have to be connected with a LAN cable to the PoE Switch. The cable “Ku/Ka UL Vector Stimulus” has to be disconnected from the front end and connected to the input of the calibration coupler. The coupled port of the coupler is connected to the TAC input with a K m-m adaptor, the output of the coupler is connected to PWS2. PWS1 is connected to the output of the TAC. The LO signal is provided by the SYN1, SYN1 has to be connected to the TAC LO input. The TAC must be configured for FWD configuration. In order to obtain the characteristic gain curve of the TAC, an MC signal with a bandwidth equal to the frequency span of the FWD link will be generated by AWG. When applied to the input of the TAC, the broadband power will be measured at the input of the TAC with the PWS2 through a coupler. The output broadband power will be measured with PWS1 at the output of the TAC in 5 MHz steps in the bandwidth of interest. The difference between the measured values from PWS1 and PWS2 is reported as FWD transmission gain characteristic of the TAC which is shown in Figure 40.

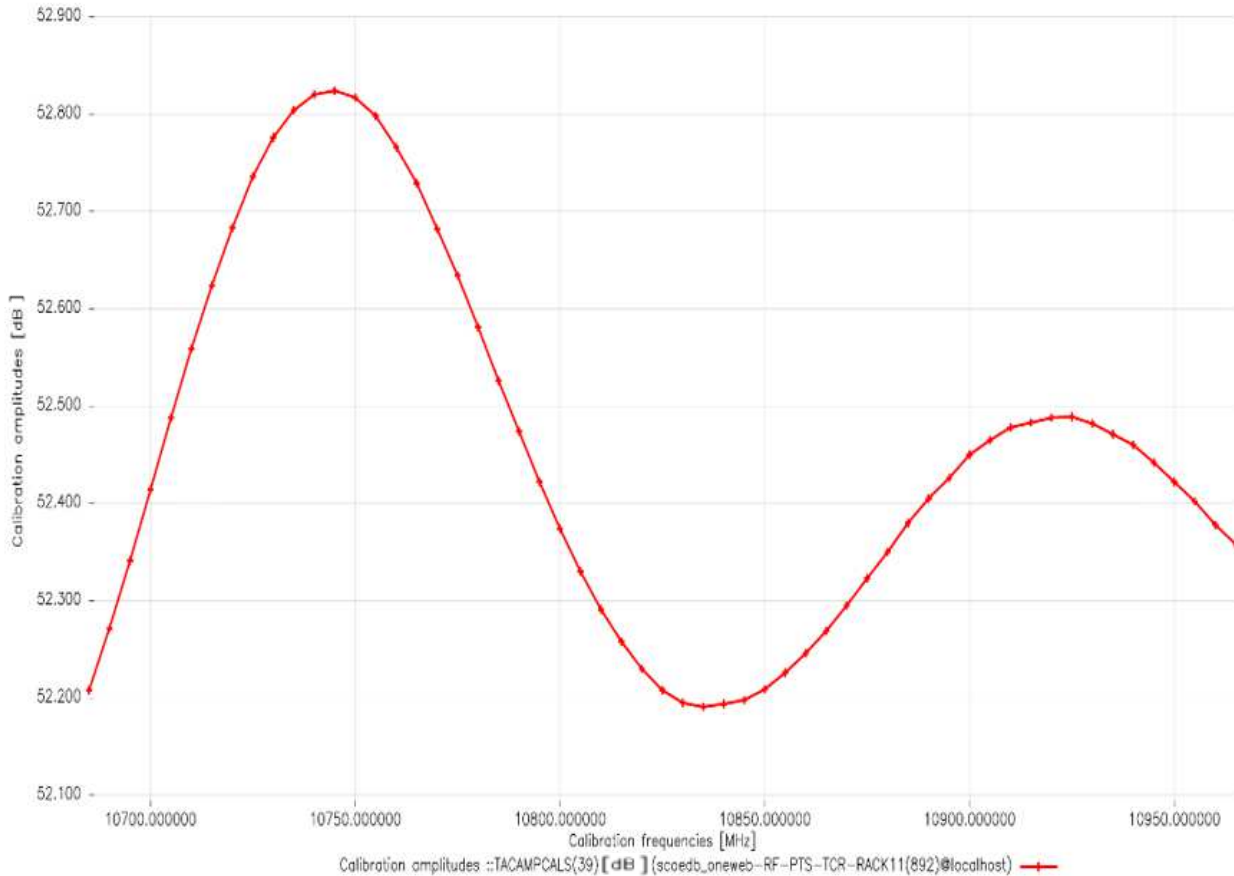


Figure 40: TAC FWD transmission gain calibration for center=10825 MHz with a span of 280 MHz.

4.1.2 Phase Calibration

The phase response is measured with a vector network analyzer (VNA) in non-frequency converting configuration, only the bandpass filters and the output amplifier are measured. The input amplifier and the mixer do not significantly contribute to group delay variations, therefore, the phase response of the signal path between the output of MIX1 and the OUT KuDL interface (see Figure 38) is reported as FWD phase response calibration.

For calibration of the TAC phase response, a vector network analyzer is needed. The VNA has to be connected to the SCOE LAN switch. An ECAL Kit is used for VNA calibration, thereafter the Port1 of the VNA is connected to the input of the ATT3 (See Figure 38) and the Port2 of the VNA is connected to the output of the TAC (OUT KuDL). The phase of the S-parameter (S21) is measured in units of degrees in the TAC calibration routine. The measured phase response results in the frequency band 10685-10965 MHz is shown in Figure 41.

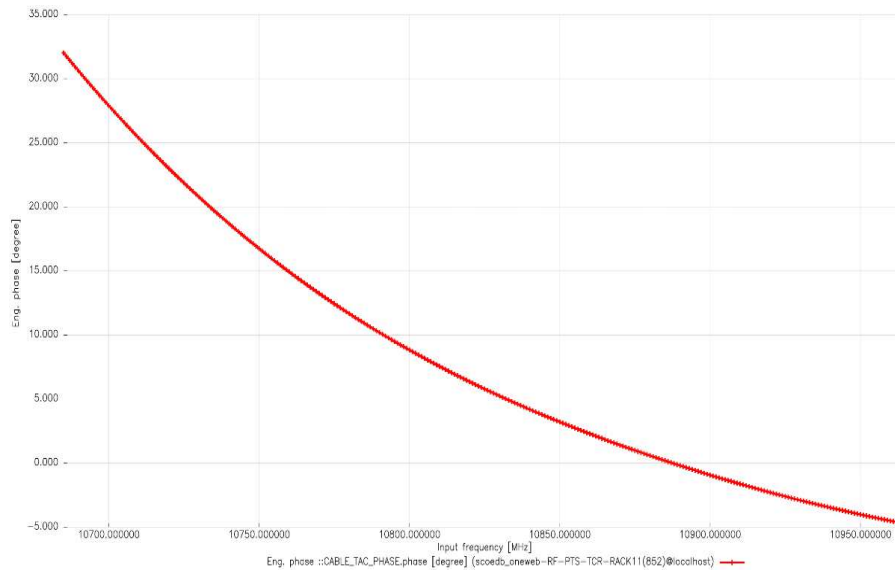


Figure 41: TAC FWD phase calibration results for the center frequency of 10825 MHz and span of 280 MHz span.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 5

Multicarrier Calibrations

5.1 Calibration of the arbitrary wave generator flatness CAL_IQ

The flatness of the I/Q modulator in the AWG can be improved by this calibration routine. This calibration is executed after a self-alignment of the AWG and before MCC. Every time a CAL_IQ routine is executed, the MCCs must be carried out again.

In a first step, the IQ flatness of the ASA is measured with a stepped CW signal over the 2 GHz bandwidth. In a second step, a 2 GHz wide chirp (in the form of an MC signal) is applied. The flatness is evaluated by calculating the difference between the MC signal trace (from step 2) and the stepped CW signal (from step 1). In a third step, the flatness list is normalized to the maximum value of the flatness list. Note that the number of steps of the stepped CW signal is equalized to the measured trace length of the MC signal.

The calibration of the AWG flatness calibrates the I/Q Impairment and I/Q gain DAC correction. An example of the I/Q calibration result is shown in Figure 42. The y-axis shows the engineering gain (Eng. gain), in this case, the gain values are normalized to an average of 0 dB. The center frequency of 2 GHz wide chirp is also normalized to 0 Hz.

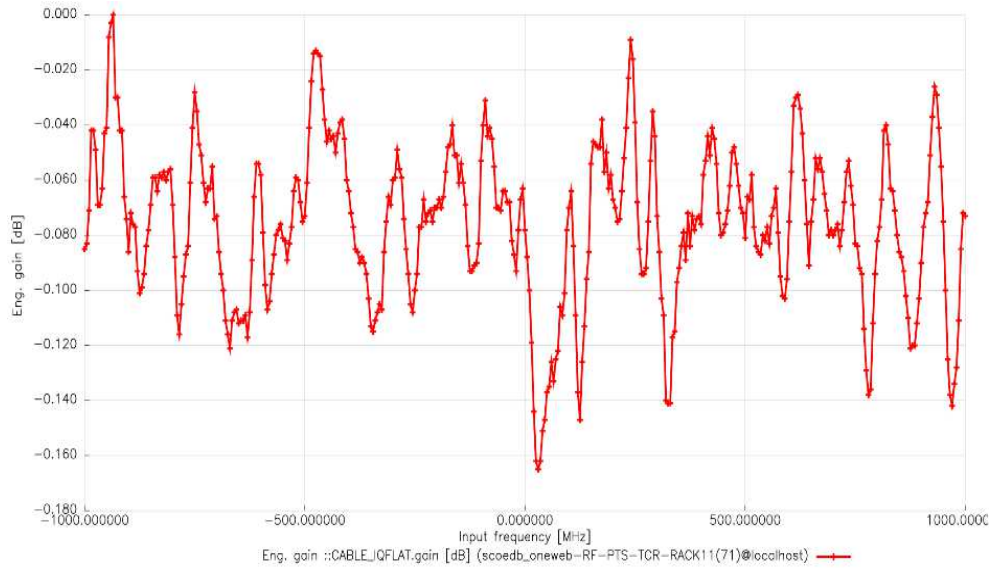


Figure 42: I/Q calibration result for compensating the I/Q response of the AWG.

5.2 Multicarrier calibrations (CAL_MCARRIER)

The input and output signals of the DUT are evaluated and compared in order to obtain the group delay and gain of the DUT with the frequency conversion. This is a challenging task. The signal at the output of the DUT in the Ku-band and the input signal in the Ka-band are measured with ASA. Note that the ASA has a different amplitude and phase responses at the stimulus in the Ka-Band and the response in the Ku band. Improved results are achieved by using a calibration data set in addition.

Amplitude- and Phase response measurements with MC stimulus are performed faster and more precisely by using MCCs. Instead of measuring the stimulus at the stimulus frequency for reference, TAC calibration data obtained from the TAC calibration routine are used. In order to define the gain and phase characteristics of the TAC, the TAC calibration routine is executed, especially in case of a DUT with frequency conversion. Improved results are achieved by using a calibration data set since ASA has different phase response characteristics at the stimulus-(Ka-Band) and the response frequencies (Ku band). It is, therefore, necessary to generate calibration data sets with a chirp signal with the same settings as intended for the measurement (bandwidth and spacing, as well as the ASA measurement configuration: YIG-filter, attenuation, reference level and so on). Some of these configurations are configurable (e.g., reference level) while others are hardcoded in the measurement routine. The calibration parameters are configurable in the GUI of the SCOE.

The MCC has to be performed after all other calibrations because this relies on the stimulus-, the response-, the IQ-, and the TAC-calibrations.

If a self-alignment of the ASA is initiated, the MCC is repeated. The setting for MCC and MCM must be identical. Otherwise, the measurement is prone to errors.

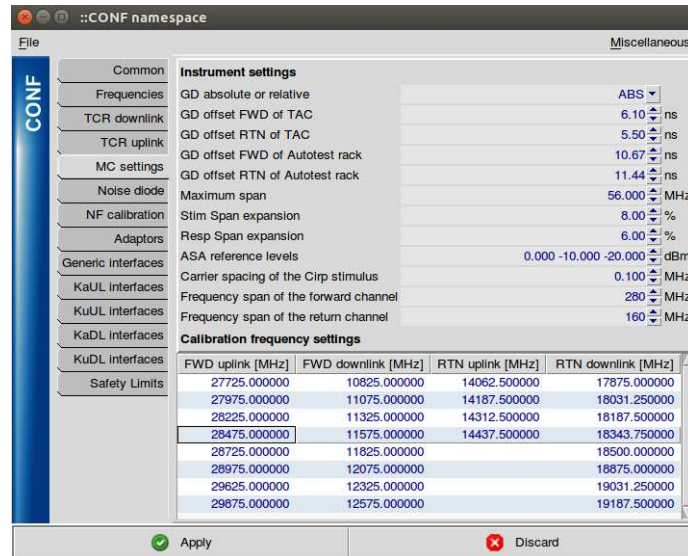


Figure 43: MC setting window in the GUI for TAC- and MCC and MCM[12].

5.2.1 Configurable parameters of the multicarrier setting window

The MCCs and MCMs settings are configurable via MC settings Tabs of CONF namespace window. These configuration parameters are utilized by either MCCs or MCMs[12].

5.2.1.1 Group delay absolute or relative

If the group delay set to “relative”, the group delay trace is normalized to 0 ns at the center of the trace. When measuring with a large single span, the “relative” setting speeds up the measurement, because re-measuring with a smaller span is skipped[12].

5.2.1.2 Group delay offset of the forward link:

These are the group delay values of the TAC for forward and return channels at the center of the frequency band (10825 MHz). Since the TAC is used as a calibration device, these values are necessary for determining the absolute group delay[12].

5.2.1.3 Maximum span

In the response measurement, the analyzer will be set to this span plus the value of “Resp span expansion”. This parameter can be set for two modes:

- 1) Span \leq 56MHz: The span to be measured is split up into several sub spans with sub span bandwidth equal to the **maximum span**. The group delay is measured absolutely with multiple measurements over each sub span. When the **maximum span** is set to be 56MHz, five measurements are executed in the MCC routine for covering a span of 280MHz.
- 2) Span = 280MHz (<318 MHz): The measurement is carried out with one single broadband sweep over the configured channel bandwidth which is defined by parameter "Frequency span of the forward channel"[12].

5.2.1.4 Stimulus span expansion

The spectrum of a chirp signal has almost 7 dB attenuation near the start- and stop frequencies. After expanding the span for the ASA by the percentage of "**Resp span expansion**", the stimulus signal span is further expanded by this parameter (e.g. 6%). This expansion reduces these amplitude variations at the start- and stop frequencies[12].

5.2.1.5 Response span expansion

Each sub span is measured with this expanded span for generating overlapping sub spans. This overlapping is required for concatenating the measurement's overall sub spans to one single measurement result[12].

5.2.1.6 Reference level of the analog spectrum analyzer

The calibration and the measurement are both performed at exactly one of these reference levels listed in the ASA reference levels field. For reasonable results, the actual signal level at the ASA should not be 30 dB lower than the reference level. The reference level should always be adjusted according to the power level of the payload. If the reference level is very different from the signal level, then the user will lose accuracy in the amplitude measurement. For reference levels $\leq +5$ dBm the attenuation of the ASA is always set to 5 dB for better return loss[12].

5.2.1.7 Carrier spacing of the chirp stimulus

The recommended value is 0.1 MHz, this corresponds to a chirp period of 10 μ s. A smaller spacing gives more lines for average, but the power per carrier is lower. Due to the limited ASA memory, the averaging carried out by the ASA is less. A wider spacing results in fewer spectral lines for calculating the group delay. When measuring with one single broadband sweep, a wider spacing (0.2 or 0.5 MHz) gives less noisy results. For more information, see Sec. §2.3.1[12].

5.2.1.8 Frequency span of the forward channel

The MCC is either performed for these frequency spans or a smaller span in case the **maximum span** is smaller than the frequency span of the forward channel. The frequency span of the FWD channel is selected in a wider range than the **maximum span**. If these values are greater than the **maximum span**, the MCC is split up into several smaller subspans[12].

5.2.1.9 Calibration frequency settings

This is a list of the center frequencies of all channels. Calibration is performed over all combinations of “FWD uplink” and “FWD downlink” frequencies. The MCC frequencies are selected from the Calibration frequency settings list.

All the measurements and tests in this document are executed and verified for our purposes only in the FWD link with a stimulus signal at center frequency 27725 MHz. This is the first Ka-band uplink channel out of 8 uplink channels defined by AOS. The response signal at center frequency 10825 MHz is the first Ku-band downlink channel out of 8 downlink channels specified by AOS. We have chosen the FWD link for the measurement which is more prone to errors due to higher operating frequency range.

It is important that MCC frequencies are identical to the stimulus and response center frequencies configured in the MCM. This ensures that the calibration and the measurement are performed at exactly the same instrument settings. The appropriate calibration data are retrieved from the database. Any changes of these parameters require a new MCC[12].

5.2.2 Multicarrier calibration routine

5.2.2.1 splitSpan

The subroutine splitSpan utilizes the stimulus center frequency (taken from calibration frequency setting list), span (taken from frequency span of the forward channel), spacing (taken from MC settings window) and **maximum span** (taken from MC settings window) and delivers a list of center frequencies and spans designed for each sub span. If the measurement span is greater than the **maximum span**, then it delivers a list of center frequencies and spans dedicated to each sub span. Otherwise, the **splitSpan** subroutine will not be used. For example, if the stimulus center frequency, span, spacing, and maximum span are set to 27725 MHz, 280 MHz, and 0.1 MHz, 56 MHz, respectively then the subroutine splitSpan outcome will be a list of center frequencies and a list of spans as listed below.

stimulus center list = (27613.0 MHz, 27669.0 MHz, 27725.0 MHz, 27781.0 MHz, 27837.0 MHz)

span list = (56 MHz, 56 MHz, 56 MHz, 56 MHz, 56 MHz)

5.2.3 Multicarrier amplitude calibrations

The amplitude at the output of the TAC from the translated chirp signal is measured by using the ASA and the downlink calibration data. The calibrated amplitude trace is then subtracted from the TAC amplitude calibration (which is retrieved from the database) and finally normalized to a total power of 0 dBm. Figure 44 shows an example of the MC amplitude calibration data normalized to 0 dBm with the stimulus center at 27725 MHz, response center at 10825 MHz and a sub span bandwidth of 56 MHz.

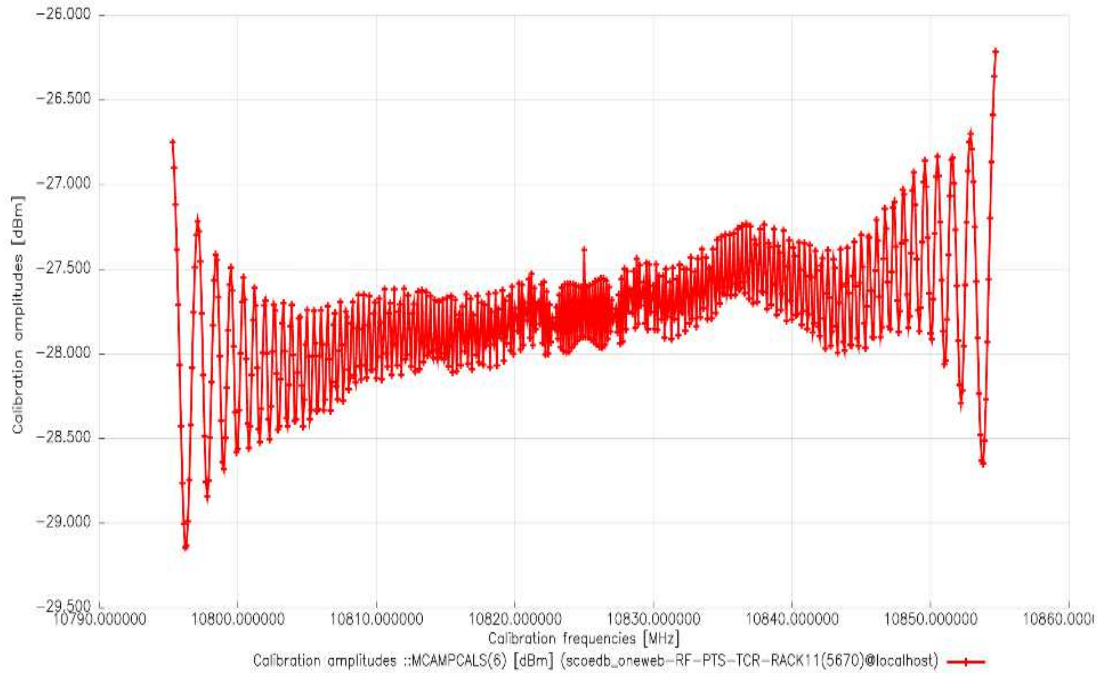


Figure 44: MC amplitude calibration with stimulus center 27725 MHz, response center 10825 MHz and bandwidth 56 MHz.

5.2.4 Multicarrier phase calibrations

In the phase measurement, as already discussed in Sec. §2.2.11, it is important to measure the phase of the DUT in a bandwidth range smaller than 80 MHz in order to meet the absolute group delay measurement requirements.

In the MC phase calibration measurement routine, a chirp signal is set up with a specific uplink center frequency taken from the MC settings window. The phase response of the DUT is measured with frequency conversion at the downlink frequency by ASA (see Figure 45).

The difference between this phase response measurement and the measured phase in TAC phase calibration is depicted in Figure 46. It is noted as MC phase calibration which is illustrated in Figure 47.

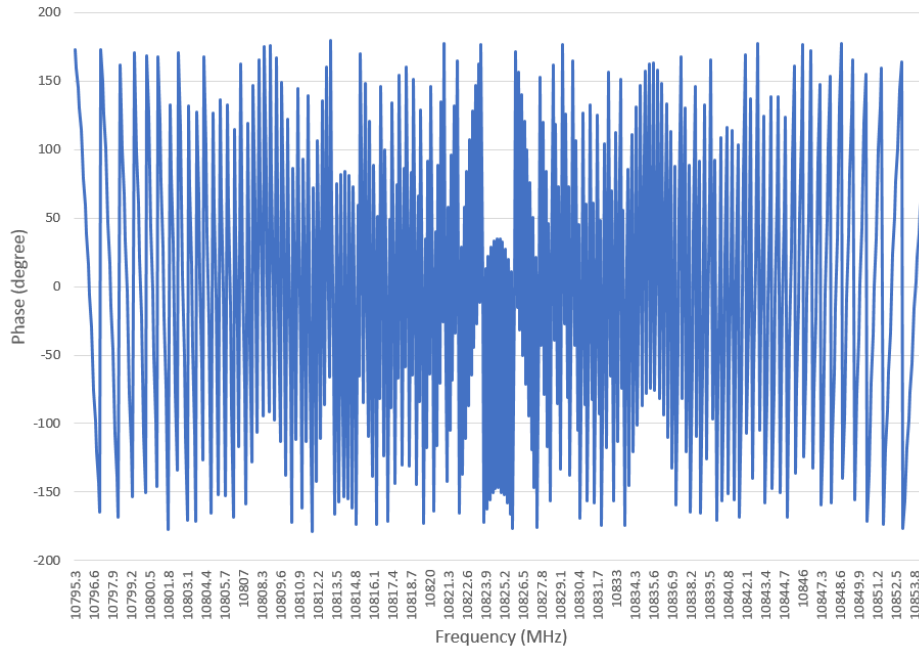


Figure 45: Measured phase response of the TAC in MCC.

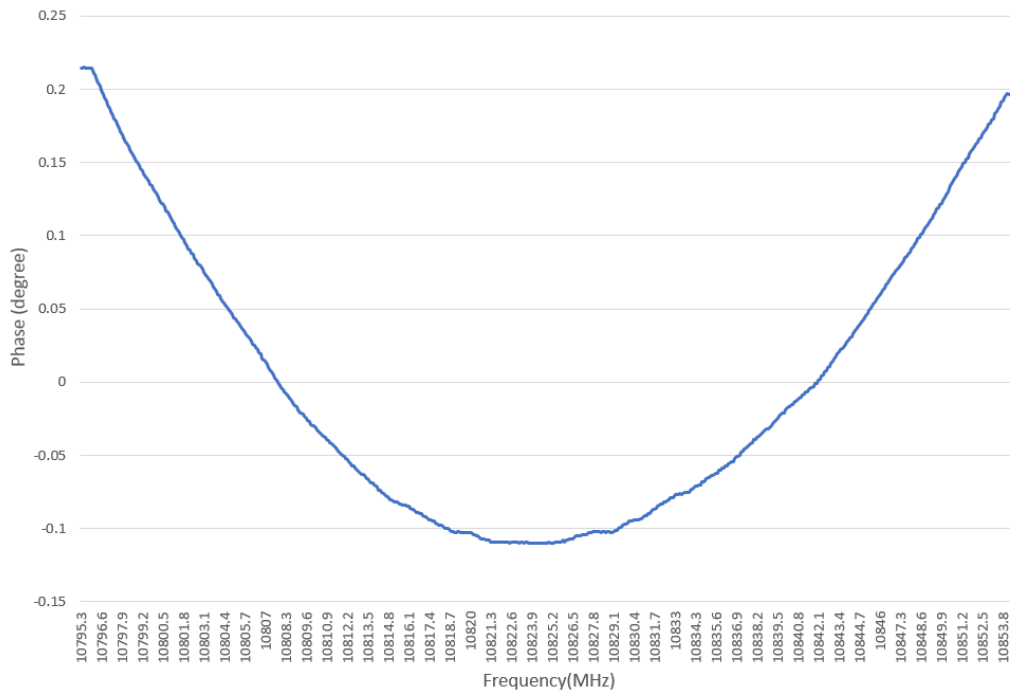


Figure 46: Processed phase response of the TAC from TAC calibration.

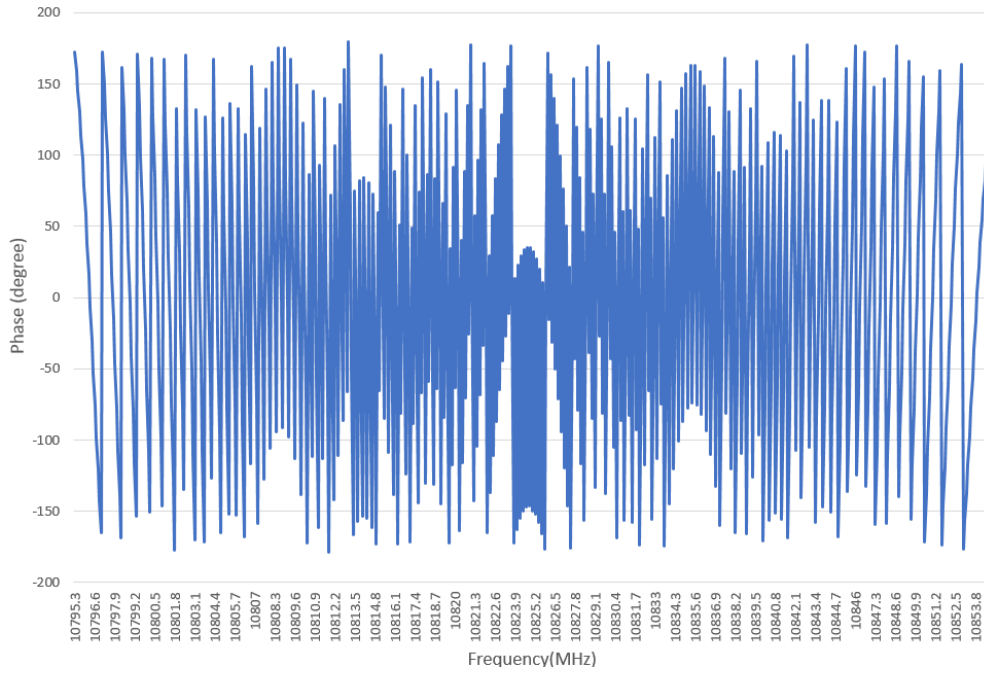


Figure 47: MC phase calibration.

Chapter 6

Multicarrier measurements

The MEAS_MCARRIER routine determines the amplitude- and phase response of the DUT by applying an MC stimulus (chirp signal) and using the I/Q analyzer option on the ASA for evaluating the response signal. This option only offers the amplitude and phase of the signal. The group delay estimate is calculated in the measurement routine.

6.1 Configurable parameters of MEAS_MCARRIER

respCenter: Center frequency of the response signal in MHz. This frequency has to be selected from the list of “calibration frequency settings” from CONF namespace.

respPath: Selected response path (e.g. KaDL1-GW1-R or KuDL1).

span: Frequency span in MHz for MCM (typically 125 or 250MHz, must be chosen smaller than or equal to the configured value from CONF namespace).

aperture: Frequency aperture in MHz for calculating the derivatives of the phase response for the group delay and of the amplitude response for gain slope calculations; has a smoothing effect. Default value = 1 MHz. Wider aperture gives smoother traces with less frequency resolution.

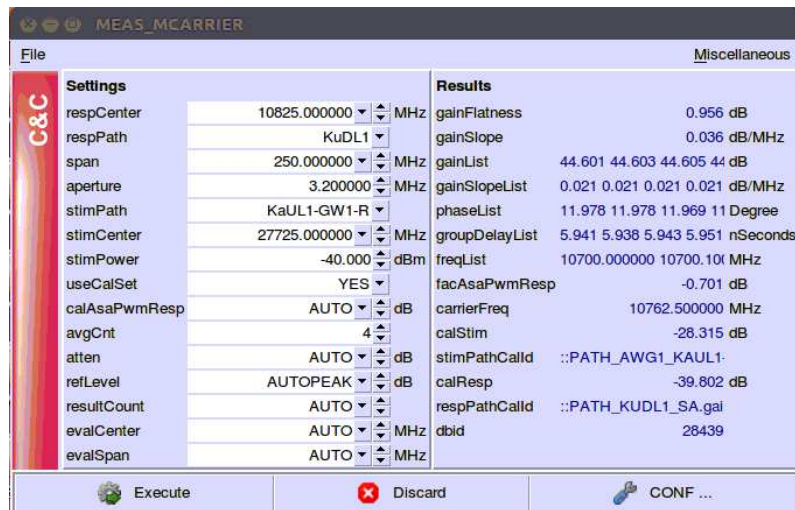


Figure 48: GUI for multicarrier measurement.

stimPath: Stimulus path (e.g. KaUL1-GW1-R or KuUL1).

stimCenter: Center frequency of the MC stimulus in MHz. This frequency has to be selected from the list of “calibration frequency settings” from CONF namespace when using MCCs.

stimPower: Power of the stimulus in dBm at the calibration plane.

useCalSet: If set to “Yes” then the stimulus is not measured again and the calibration data is used instead. This is faster and more precise and, therefore, recommended. Of course, this requires appropriate calibration beforehand. If set to “No” then the calibration data are not used, and the stimulus signal is measured in the internal test loop as a reference. This setting is slower and results in a higher ripple, but this is more flexible without prior need for calibration.

calAsaPwmResp: The absolute gain is measured with a CW carrier at two frequencies and a calibration measurement of the ASA against the power sensor is performed. The default value is “AUTO”. An <value> in dB = Use this value derived from previous measurement result (or set to 0); this is not so precise but somewhat faster because the calibration against the power sensor is skipped.

avgCnt: Number of averages, defaults to 1. At low output levels or poor S/N ratio the averaging count can be increased for noise suppression. Values for **avgCnt** up to 8 are recommended.

atten: ASA attenuator setting in dB, defaults to “AUTO”. The attenuation will be set to the lowest possible value for better signal-to-noise ratio, but not lower than 5 dB for better input return loss of the ASA. A <value> in dB = The measurement is done with this setting. If useCalSet=true is selected, this value should be “AUTO” or 5 dB, otherwise, the calibration will not match the measurement.

refLevel: “AUTOPEAK” = The reference level of the ASA is set automatically in the broadband acquisition mode (BW = 320 MHz). This ensures the detection of the highest signal peaks within this bandwidth and is the recommended method for broadband noise-like signals. “FSW AUTOLEVEL” = The built-in auto-leveling method in the spectrum mode of the analyzer is used; recommended for narrowband signals. “AUTO PWS” means the input power is measured in broadband mode with the power sensor; the

crest factor should be low. A <value> in dB = The reference value of the ASA is set to this value. If **useCalSet=true** is selected, the **refLevel** should be automatically selected or one of the values listed in “ASA ref levels” in the CONF namespace, otherwise no suitable calibration can be found.

resultCount: The number of measurement values in the result lists is decimated to this number. The useful range is a minimum of 10. If set to “AUTO” then this means that the maximum value is chosen (=span/carrier spacing).

evalCenter: Center frequency in MHz of the evaluation bandwidth for gain slope and gain flatness; default = center

evalSpan: Evaluation bandwidth (symmetrical to **evalCenter** frequency); default = span. For calculation of the gain slope and gain flatness, only measurement values inside this **evalSpan** will be considered[12].

6.2 Results of multi-carrier measurements

gainFlatness: Max gain – min gain over evaluation bandwidth in dB.

gainSlope: Maximum gain slope over evaluation bandwidth in dB/MHz.

gainList: Data set of measured gain values in dB; list length = resultCount.

gainSlopeList: Data set of computed gain slope values in dB/MHz, derived from the **gainList**, list length = resultCount.

phaseList: Data set of measured phase difference values in Degrees, converted to a continuous increasing phase list, list length = resultCount.

groupDelayList: Data set of the computed group delay response in ns, derived from the **phaseList**, list length = resultCount.

freqList: List of measured frequencies in MHz for gain, phase and group delay lists, listlength=resultCount.

facAsaPwmResp: If calAsaPwmResp = “AUTO”, this is the measured correction value of the ASA-PWM calibration. Otherwise, it is identical to the configured calAsaPwmResp value.

carrierFreq: measured frequency value of the response signal. For determination of the exact gain value, a CW stimulus at stimulus center frequency with **stimPower** is applied, and the amplitude and frequency of the response signal are measured.

calStim: calibrated gain values at the stimulus center of the selected stimulus path.

calResp: calibrated gain value at the response center of the selected response path[12].

6.3 Multi-carrier measurements routine

The main measuring loop applies a chirp stimulus with a center frequency of stimulus center, a power level of **stimPower** (specified in the GUI of MCM) and with a bandwidth of span to the DUT. If the span is smaller than maximum bandwidth then span is taken as the bandwidth of the chirp waveform. Otherwise, the subroutine splitSpan divides the span into smaller sub spans.

6.3.1 Gain (magnitude flatness) measurements

The gain is defined as the relation between the measured magnitude and the reference (calibration) magnitude. It describes the transmission behavior of magnitude versus frequency of the DUT,

$$Gain(f) = \frac{Magnitude_{output}}{Magnitude_{input}}$$

Equation (18): Gain definition in the frequency domain.

The response amplitudes of the DUT stimulated by the chirp signal are measured with the IQ Analyser measurement routine. The measured amplitudes are calibrated with the response path calibration data to obtain high precision output power level estimates for the DUT. The subtraction of the calibrated response amplitude (in dBm) and the MCC stimulus amplitude level (in dBm).

$$calibrated\ response\ amplitude(dBm) = response\ amplitude(dBm) + path\ attenuation$$

Equation (20): Calculation of the response amplitude level at the DUT output port.

$$gain(dB) = calibrated\ response\ amplitude(dBm) - calibrated\ stimulus\ amplitude(dBm)$$

Equation (20): Calculation of the gain as difference of response amplitude level and stimulus amplitude level.

6.3.2 Group delay measurements

The group delay is a measure of the device's phase distortion. It is a little bit difficult to consistently interpret the group delay response as the transit time of a narrowband signal through an RF device. The definition of group delay is based on the derivative of the device's phase response with respect to frequency.

$$\tau(f) = -\frac{1}{2\pi} \times \frac{d\varphi_{\Delta}(f)}{df}$$

Equation (19): Group delay definition in general.

Since the phase is ambiguous and only defined up to a constant which must be a multiple of 2π , it is possible to unwrap the phase response such that $\frac{d\varphi_{\Delta}(f)}{df}$ becomes a smooth function without discontinuity. This

unwrapping operation needs to be carried out before the evaluation of the derivative in the group delay definition.

Where:

$$\varphi_{\text{delta}}(f) = \text{unwrap}(\varphi_{\text{meas}}(f) - \varphi_{\text{cal}}(f)) \quad \text{Equation 20: calculate phase delta function.}$$

The phase and frequency responses of the chirp signal are measured with the IQ analyzer measurement routine. The difference (in degrees) between the measured phase at ASA and the calibration phase from MCC is calculated as the measured phase response of the DUT. The group delay estimate (in ns) is calculated by substitution of measured phases and frequency response of the DUT in Equation (19).

6.3.2.1 Linear phase-shift component

The linear phase-shift component represents average signal transit time and is attributed to the electrical length of the DUT[13].

6.3.2.2 Higher-order phase-shift component

The higher-order phase-shift is a source of signal distortion which represents variations in transit time for different frequencies[13].

6.3.2.3 Absolute vs relative group delay

In a practical RF circuit like a cable, the velocity of the signal is lower than in vacuum, and the delay due to the electrical length is higher than one would expect based on the length of the cable, the measurement of this delay is called absolute group delay. In practice, the absolute group delay is important in ranging applications where the distance between two points shall be measured. However, the absolute group delay may be significant, this delay affects all frequency components in the same way and does not lead to a change in the signal shape.

In a correct relative group delay measurement, the group delay trace over frequency should be around 0 seconds. The group delay is calculated relative to the reference signal and the constant delay caused by DUT is eliminated[14].

6.3.3 Measurement Results

The frequency span of the forward channel designated by OneWeb is set to be 280 MHz. In the following, a 280 MHz broadband chirp signal is applied to the DUT. The amplitude and phase responses of the DUT are measured with one single sweep of ASA for calculating the gain and the group delay traces, respectively. The results of multi-carrier transmission gain and absolute group delay measurements by applying chirp signals with a bandwidth of 280 MHz are shown in Figure 49 and Figure 50.

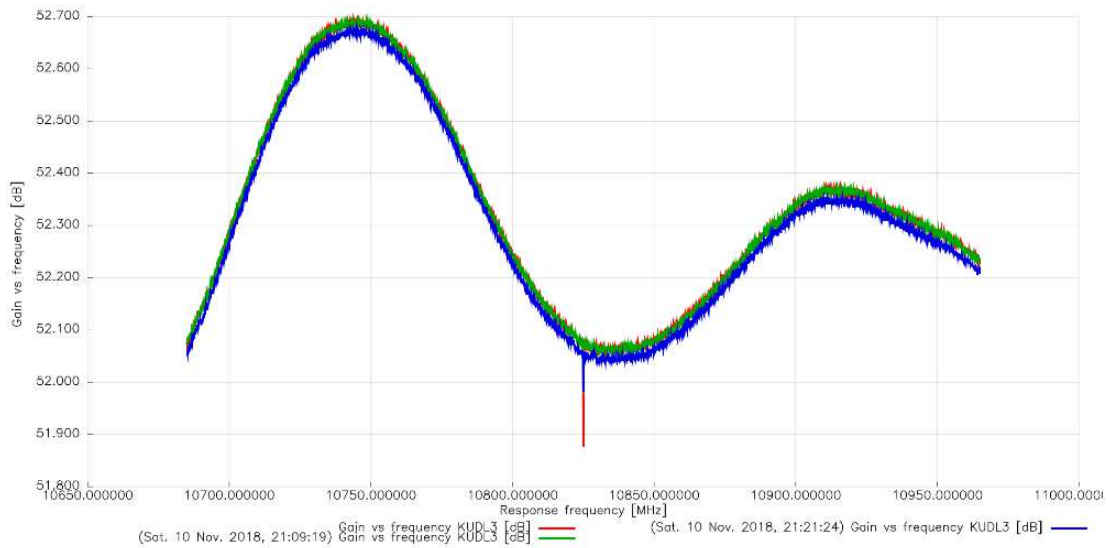


Figure 49: Broadband transmission gain measurements.

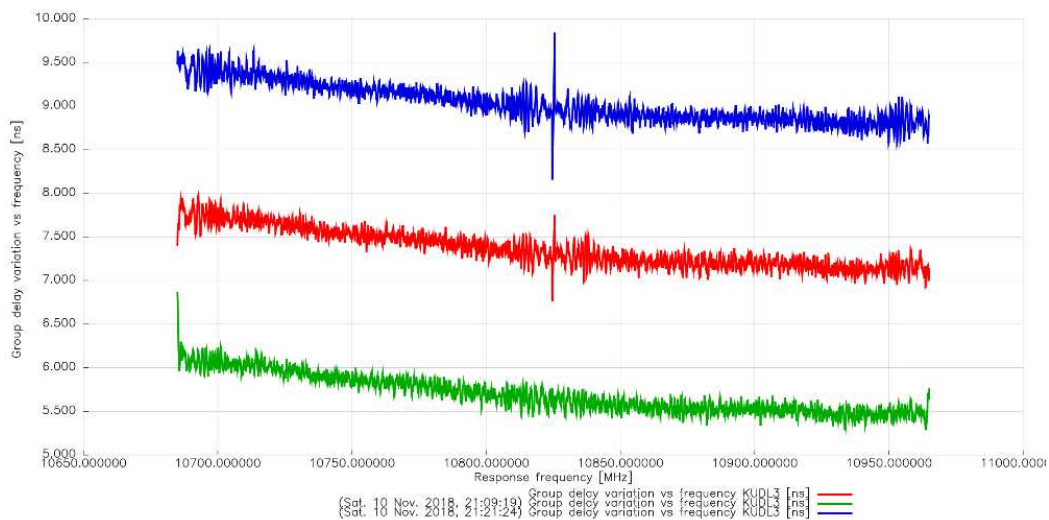


Figure 50: Broadband group delay measurements.

As shown in Figure 49 and Figure 50, the measured gain shows similar behavior to the one in Figure 19. In broadband measurement (maximum span=280 MHz) by repeating the same measurement, the gain result is always the same. However, this is not true for group delay measurement. In other words, the broadband phase measurements display different absolute group delay variation for the same DUT. Obviously, this is not correct. This behavior results when the bandwidth is larger than 80 MHz because for such large bandwidths the trigger offset cannot be inquired accurately as discussed in Section 2.2.11. Hence, the whole measurement bandwidth should be split into smaller sub spans (less than 80 MHz) in order to be able to measure the phase response and group delay of the downlink signal accurately.

6.3.4 splitSpan

The routine splitSpan is used for MCM and MCC. This routine is documented in Sec. § 5.2.2.1.

Performing MCM while splitSpan subroutine is used in MCM routine results in Figure 51. All chirp signals are set up with spectrum bandwidth of 56MHz, the response amplitude and phase are measured with IQ analyzer modus of ASA with IQ analyzer measurement routine at 56 MHz analysis bandwidth. The results are demonstrated in Figure 51 and Figure 52.

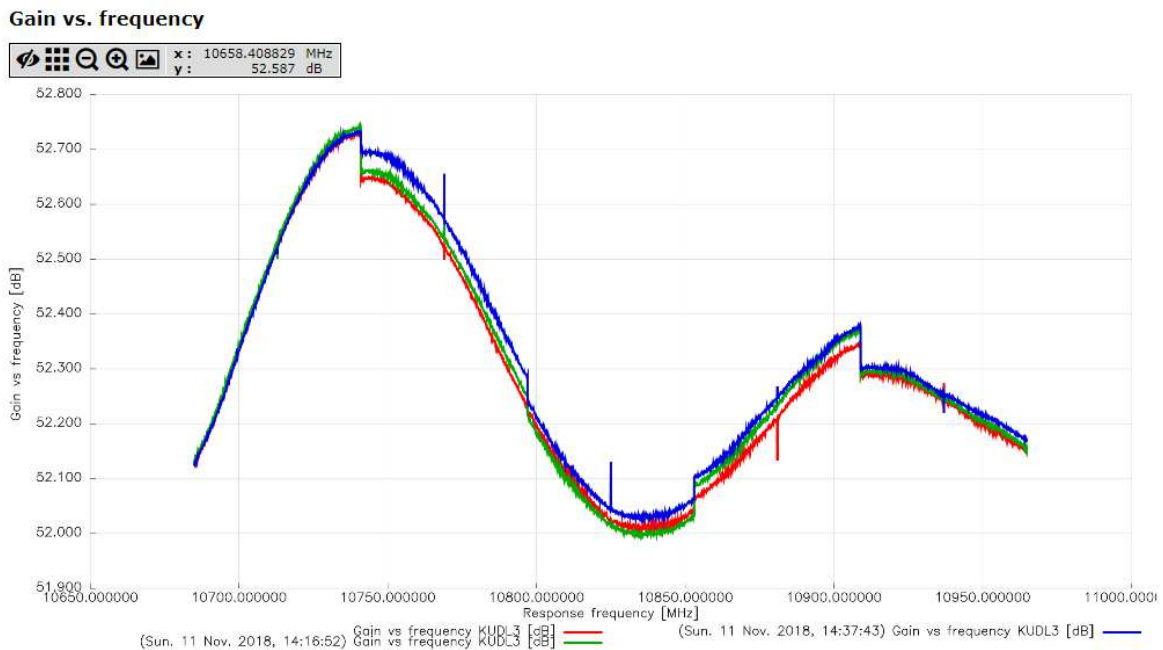


Figure 51: Gain measurement with no stimulus and response expansion.

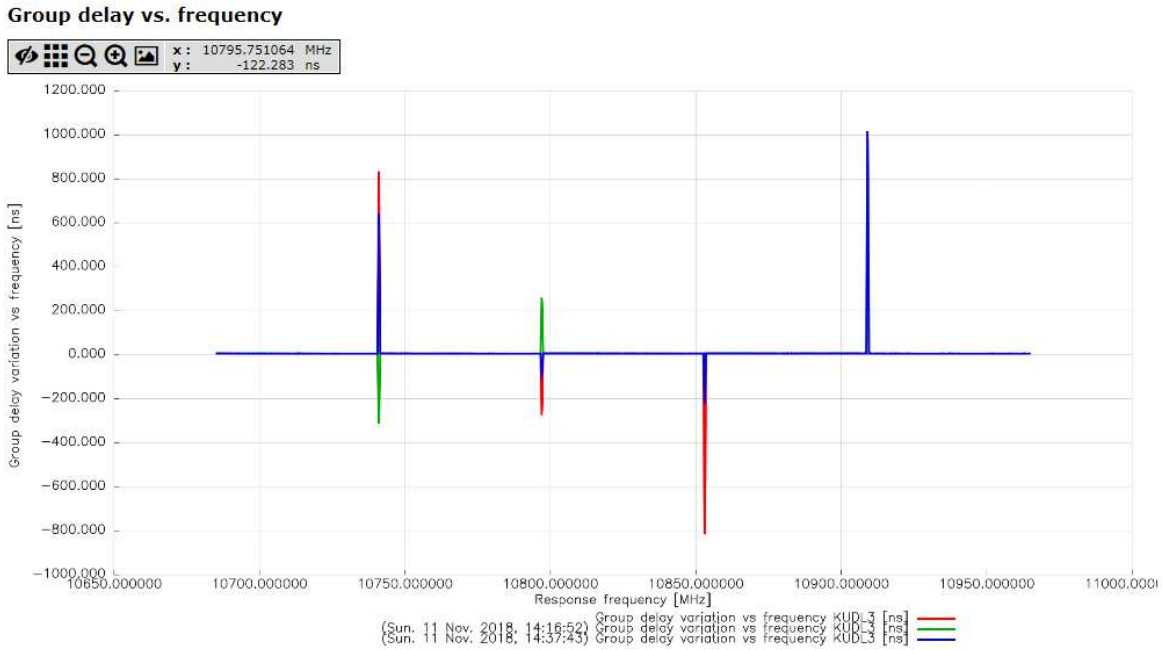


Figure 52: Phase measurement with no stimulus and response expansion.

As is evident from Figure 51 and Figure 52, there are several stepwise discontinuities in the measured trace. These step discontinuities are located exactly at the junction of two sub span. They are due to the fact that a new setup of the chirp signal is devoted to a sub span result in different amplitudes and phases. Indeed, the concatenation of the amplitude- and phase responses for each sub span causes the breaks in the measured phase. E.g. if the phase trace for the first sub span ended with -170° and the phase trace for the second sub span starts with 180° , then a large difference between these two values causes a spike or peak in our group delay measurement. Additionally, as shown in Figure 53, the applied chirp waveform does not have a constant amplitude over frequency in the whole bandwidth, especially at the start and stop frequencies. The amplitude variations at the edges of the sub span can sometimes be up to almost 7 dB. These cause a ripple in the gain measurement around the junctions between two sub spans.

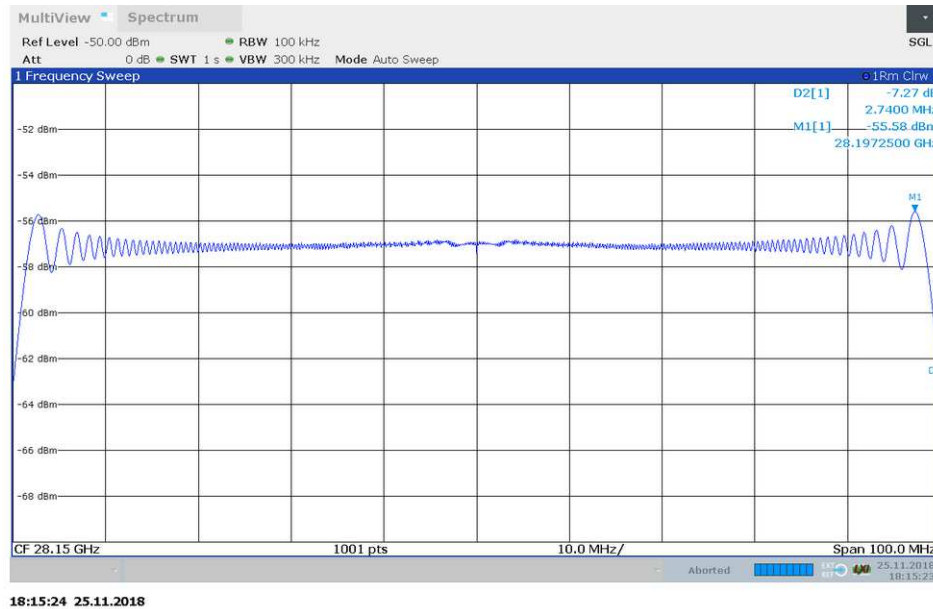


Figure 53: Chirp spectrum sideband.

The concept to mitigate this ripple is to slightly expand the chirp bandwidth of each sub span. In this way, the chirps overlap in adjacent regions which allow stitching together the individual measurements in adjacent sub spans in a continuous manner. This chirp bandwidth expansion also mitigates the amplitude variation at the band edges of the chirp waveform. The chirp bandwidth expansion factor is expressed as a percentage of Resp Span Expansion, see Section § Sec. \$Response span expansion5.2.1.5.

6.3.5 Stimulus and response expansion

The span of the response measurement is stretched by the “Resp Span Expansion” percentage to create an overlapping frequency range for stitching together the sub spans, For example, if Resp Span Expansion is set to 6 percent and we choose 56 MHz maximum bandwidth then the response span is calculated to be 59.4 MHz based on Equation (21).

$$response\ span = round(span \times (1 + \frac{Resp\ Span\ Expansion}{100}))$$

Equation (21): Response span expansion.

Where:

The span could be any of spans in the span list created by the **splitSpan** subroutine.

The span of the stimulus signal is expanded by the “**Stim Span expansion**” percentage to set up a wider chirp signal that excludes the sideband of the chirp signal at the lower and upper edges of the spectrum in the response measurement. E.g. with 8 percent stimulus expansion and 59.4 MHz response span, the stimulus span is calculated to be 64.2 MHz by Equation (22).

$$stimulus\ span = round(response\ span \times (1 + \frac{Stim\ Span\ expansion}{100}))$$

Equation (22): Stimulus span expansion.

As shown in Figure 54, the chirp signal occupies a bandwidth of 64.2 MHz (stimulus span) but only 56 MHz bandwidth of the spectrum is used for the measurement in which the chirp signal amplitude is almost flat.

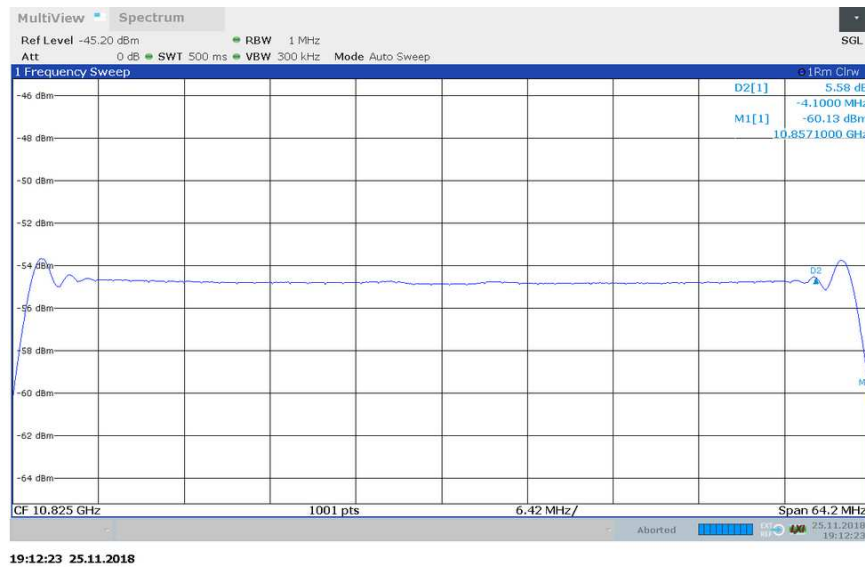


Figure 54: Stimulus Span expansion excludes the invariant part of the amplitude.

After applying stimulus and response expansion the gain curve looks more coherent at the junction of sub spans as demonstrated in Figure 55 and Figure 56. There are still some spikes at the center frequency of each individual sub span in both gains- and group delay measurement.

Gain vs. frequency

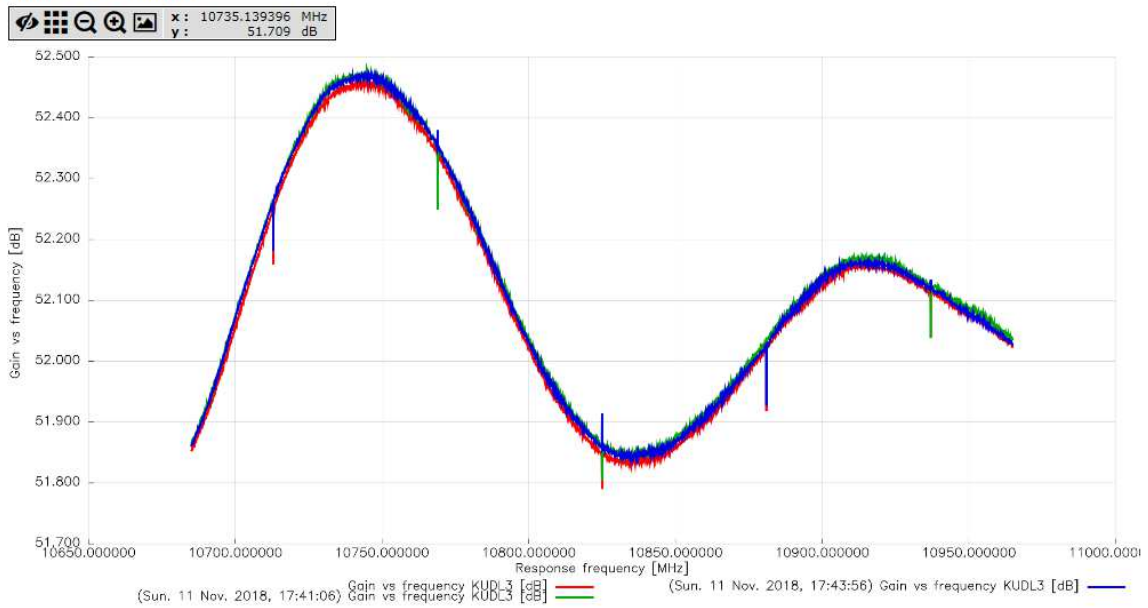


Figure 55: Gain measurement with stimulus and response expansion.

Group delay vs. frequency

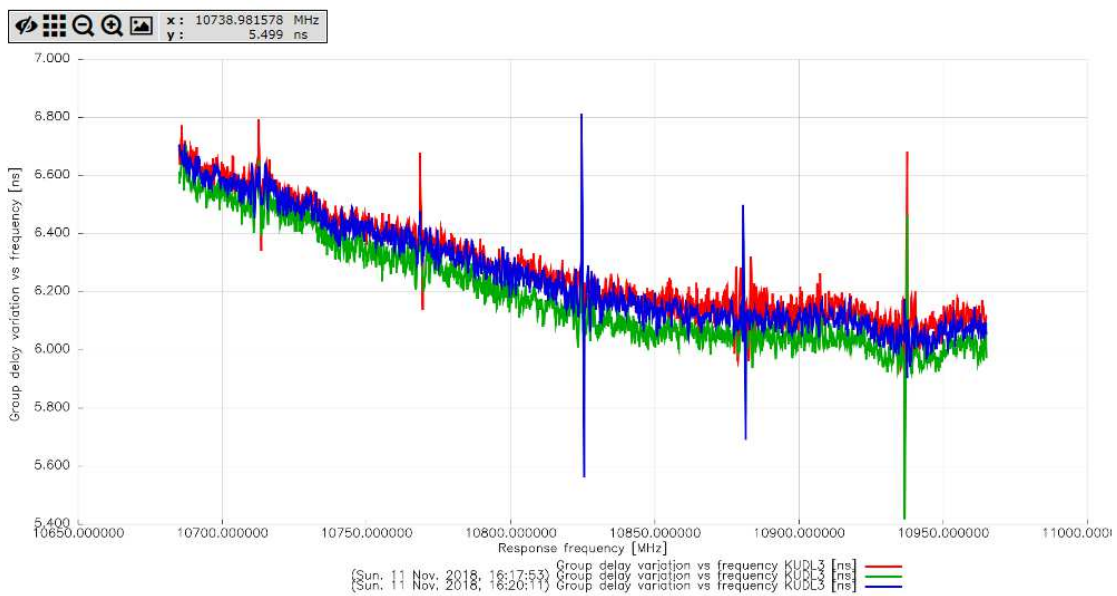


Figure 56: Group delay measurement with stimulus and response expansion.

Real-world RF devices do not behave as ideal as in the corresponding idealized simulation or theoretical calculation. Particularly, RF devices produce (among others) the following undesired effects:

- A DC Offset results in spikes at the center frequency in the amplitude measurement.
- An IQ Imbalance results in spikes at the center frequency in the phase measurement.

There is an option in the IQ modulator settings of the Rohde and Schwarz SMW200A instrument which is called **Adjust I/Q Modulator Current Frequency** that starts the adjustment for the I/Q modulator for the currently set frequency and baseband gain. The I/Q modulator is adjusted according to the carrier leakage, I/Q imbalance, and quadrature. This option estimates these effects and removes them via digital or analog signal processing.

The **Adjust I/Q Modulator Current Frequency** option is applied to the measurement results shown in Figure 55 and Figure 56. Note that there are still some spikes at the center frequency of each sub span in both the amplitude and phase measurements. The spike at the center of each sub span is an artifact that stems from the DC offset. DC offset or DC coefficient is the mean value of the waveform. If the mean value is zero, there is no DC offset. A waveform with no DC offset is known as a “DC-balanced” or “DC-free” waveform. If a spike is visible, then this indicates that the mean value of the signal is not zero.

6.3.6 Trace smoothing and remove outlier

(Software-based) smoothing in the frequency domain is a way to remove anomalies in the trace that may distort the results. The smoothing process over frequencies is based on a moving average over the complete measurement span. The number of samples included in the averaging process (the aperture size) is variable and is a percentage of all samples in the trace.

6.3.6.1 Trace smoothing for the transmission gain measurements

The trace smoothing routine smooths the gain trace after the amplitude is measured, after the data is analyzed, and finally written to a trace file. Thus, smoothing is really just a visual enhancement for the display of the trace, not of the data itself. The “smoothTrace” function applies a smoothing factor that depends on **spacing**, **aperture size (in MHz)** and the **number of samples** from the gain trace. In Figure 57 the whole trace with spikes is improved in comparison to Figure 55 by applying trace smoothing to the gain measurement. The functionality behind the smoothTrace function is summarized in the following Equation (23).

$$y'(s) = \frac{1}{n} \left(\sum_{x=s-\frac{n-1}{2}}^{x=s+\frac{n-1}{2}} y(x) \right)$$

Equation (23): Linear trace smoothing[15].

Where:

S: Sample number

y(s): Magnitude or gain of the samples

x: Sample offset from s

n: Aperture size

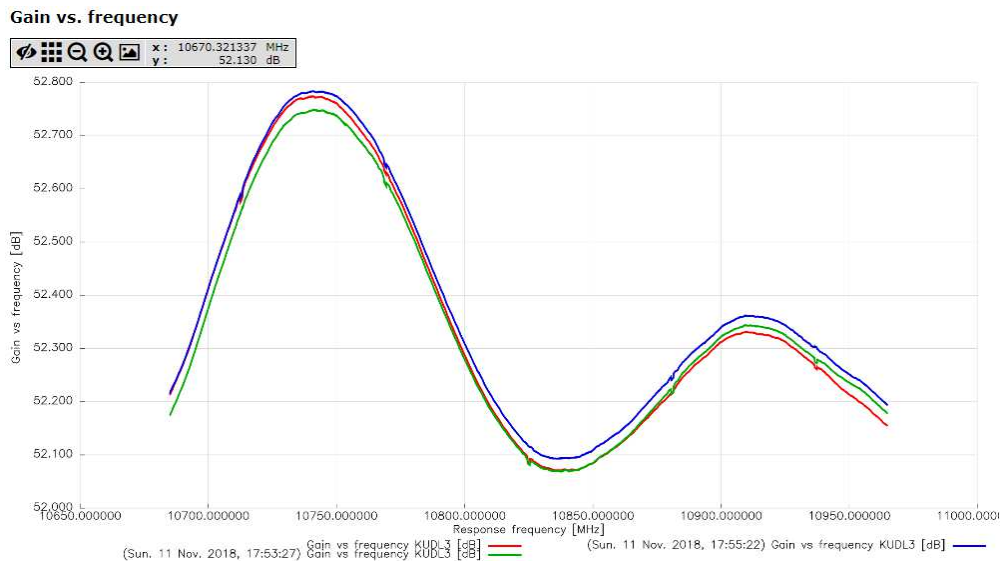


Figure 57: Trace smoothing applied to the transmission gain measurements.

6.3.6.2 Remove outlier from phase measurements

The Remove outlier from phase measurement subroutine first searches for outliers in the phase response of the signal and then replaces the detected outliers by (linear) interpolation between the two adjacent values. Figure 58 shows how the group delay measurement is improved in comparison to Figure 56 by applying the Remove outlier subroutine.

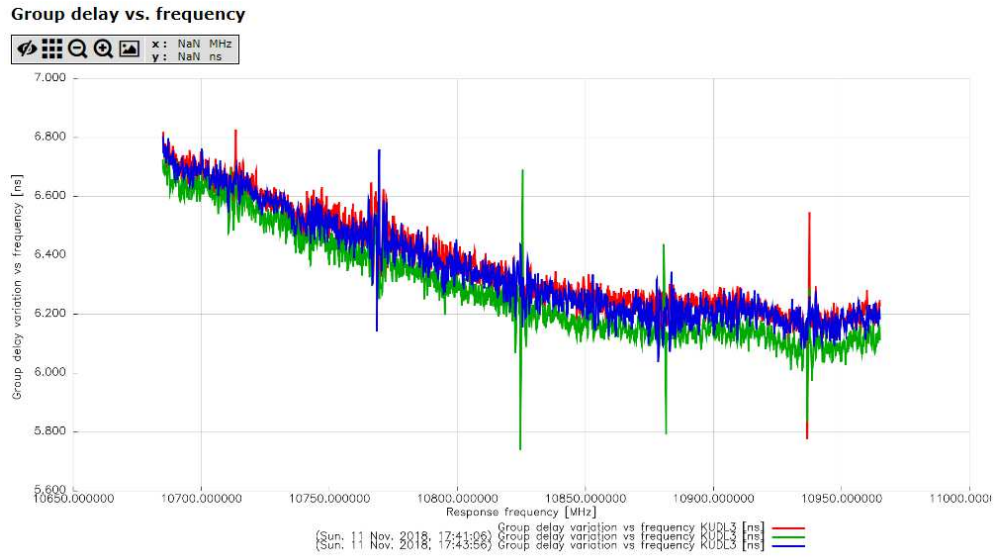


Figure 58: Remove outlier applied to the group delay measurements.

6.3.7 Eliminating DC offset and IQ imbalance

As is evident from Figure 57 and Figure 58, there are still visible spikes at the center frequencies of the individual sub spans in the gain and group delay measurement curve. The spikes are more dominant in the group delay curve. These spikes are rooted in a large variation at the center of phase- and amplitude responses versus frequency related to each sub span. In order to present the trace flatter and smoother, the center frequency measurement values for the amplitude and phase responses for each sub span are substituted by interpolation between nearest neighbors.

The spikes due to DC offset and IQ imbalance are eliminated by averaging between the nearest neighbors to the center and substitute. The operating function is noted as Equation (24) and Equation (25)

$$gain_{center} = \frac{gain_{center-1} + gain_{center+1}}{2}$$

Equation (24): Eliminating DC offset by interpolation.

$$phase_{center} = \frac{phase_{center-1} + phase_{center+1}}{2}$$

Equation (25): Eliminating IQ imbalance.

Where:

$gain_{center-1}$ =The gain of one element before the center in the gain trace

$gain_{center+1}$ =The gain of one element after center in the gain trace

$phase_{center-1}$ =The gain of one element before the center in the phase trace

$phase_{center+1}$ =The gain of one element after center in the phase trace

The MCM is repeatedly executed but this time the spikes at the center caused by DC offset and I/Q-imbalance are removed with help of interpolation, the outcome is depicted in Figure 59 and Figure 60.

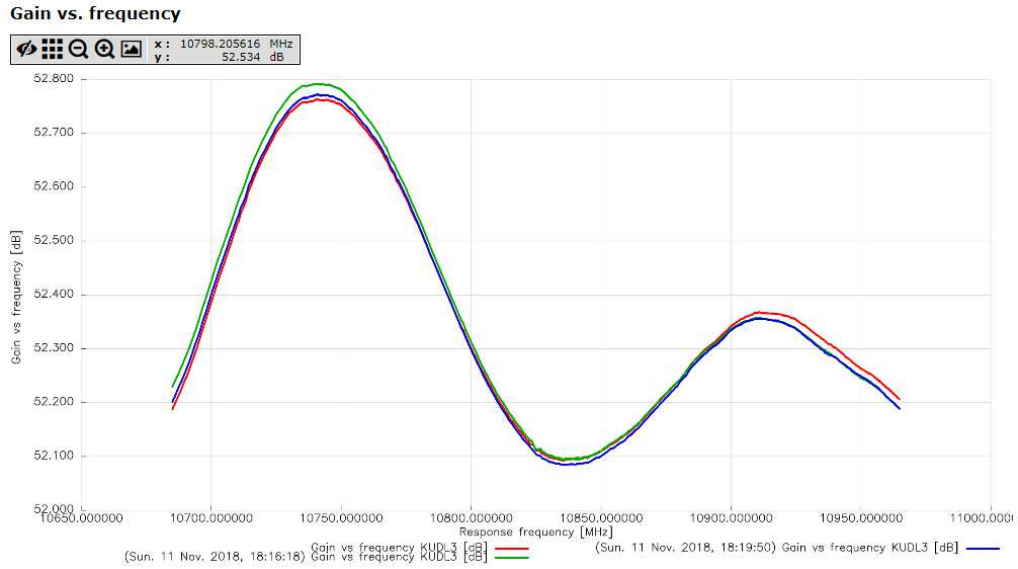


Figure 59: Center of sub spans substituted by interpolation in gain measurement.

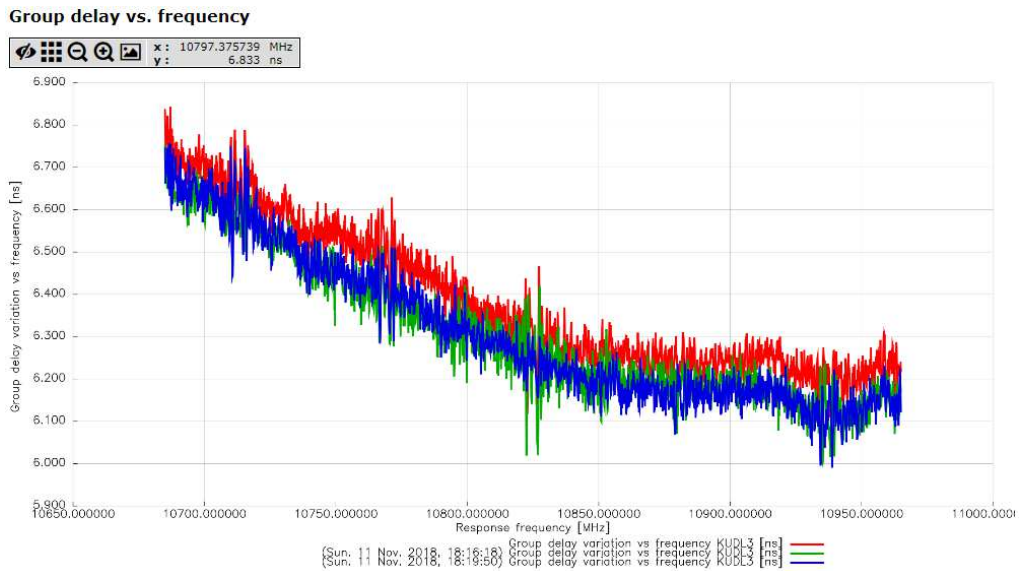


Figure 60: Center of sub spans substituted by interpolation in the group delay measurement.

6.3.8 Multicarrier gain calibration using internal loop

Previously, it was mandatory to execute the MCC prior to the MCM. The new idea is to perform calibration and measurement simultaneously. Now, it is done by implementing the calibration in the measurement routine with some limitations. As previously discussed, the broadband MCM is split up into five sub spans. The MCM and the MCC share a common measurement procedure. The main difference is that in the calibration there is no group delay measurement needed, instead, the phase will be extracted.

It is very time-consuming to execute the calibration and measurement for each sub span every time. A better solution is to run the calibration just once separately and utilize its result as the reference for all subsequent MCMs. In this way, the MCM and calibration are separated in order to save measurement time.

In Figure 61 the block diagram of RackMount and FrontEnd in OneWeb RF-SCOE are illustrated.

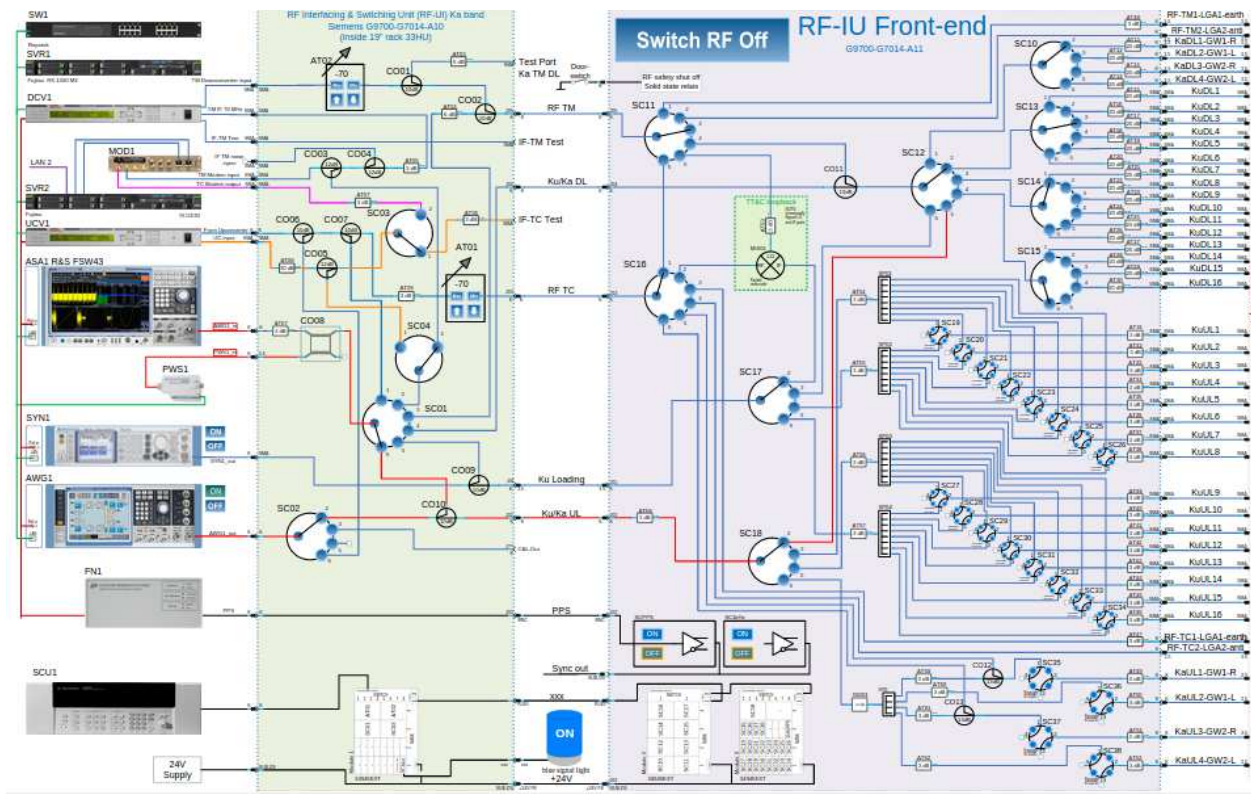


Figure 61: RackMount and FrontEnd block diagram[12].

The idea of executing MCM without MCC is realized by using the internal loop (AWG1-ASA1) through SC01 and SC02 which is highlighted in Figure 61 in red color.

There are different methods to run the MCC and use it as a reference for MCM.

6.3.8.1 Calibration in the Ka-band

The stimulus signal generated by AWG is defined in the time domain as a chirp waveform by a vector of I/Q samples (also called “I/Q pattern”) and the sample rate. The I/Q pattern is calculated by a procedure which is called calcChirp.

After the stimulus path in the calcChirp procedure has been defined, then the gain ripple of the stimulus path is read from the calibration values. Next, the inverse gain ripple is multiplied into the I/Q pattern such that the chirp signal magnitude stimulating the input of ASA1 is pre-distorted. Due to some imperfections in the ASA and AWG, the pre-distortion is not perfect.

Pre-distortion of the stimulus signal is very important. Otherwise, the measured chirp signal at ASA could not be used as a reference for the MCM.

We have already calibrated the internal loop path in the same way as the MCM. A chirp signal in the Ka-band with the same spacing, start and stop frequencies as the MCM is applied to the internal loop. The amplitude ripple of the internal path is already calculated into the pre-distorted chirp waveform. However, no frequency translation is implemented in our reference measurements. The measured amplitude response of the DUT is compared to the reference calibration amplitude measured from the internal loop in Ka-band and the difference in decibels is assumed as the gain of the DUT. In **Error! Reference source not found.**, the green curve shows the MC transmission gain measurement using the internal loop calibration at Ka-band. The red curve shows the MC transmission gain measurement using the MCC.

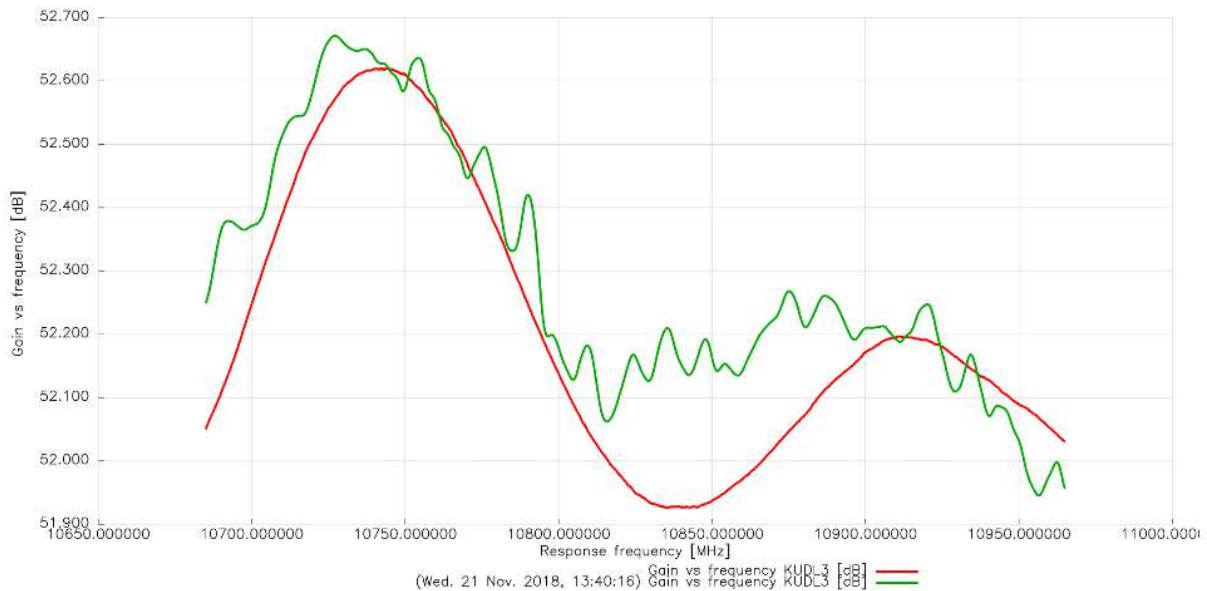


Figure 62: Multicarrier transmission gain measurements using internal loop calibration in the Ka-band as reference.

6.3.8.2 Calibration in the Ku-band

The same method mentioned in section §6.3.8.1 is implemented for the Ku-band. During MCMs the response amplitudes and phases are measured at IQ-Analyser of the ASA in Ku-band. Therefore, it makes more sense to perform the calibration in the Ku-band instead of in the Ka-band. The ASA frequency response is different in the Ku-band and Ka-band. Therefore, we try to measure the DUT's amplitude responses at the translated frequency in the Ku-band. Simply calibrating for the Ka-Band and executing the measurement in the Ku-band results in an unavoidable error in the ASA amplitude measurements. The ASA measurement uncertainty is avoidable by calibrating in the Ku-band and also measuring in the Ku-band. However, the AWG behaves differently through setting up the chirp signal in the Ku-band for calibration and applying the chirp signal in the Ka-band for measurement. Therefore, the AWG uncertainty in this method is unavoidable.

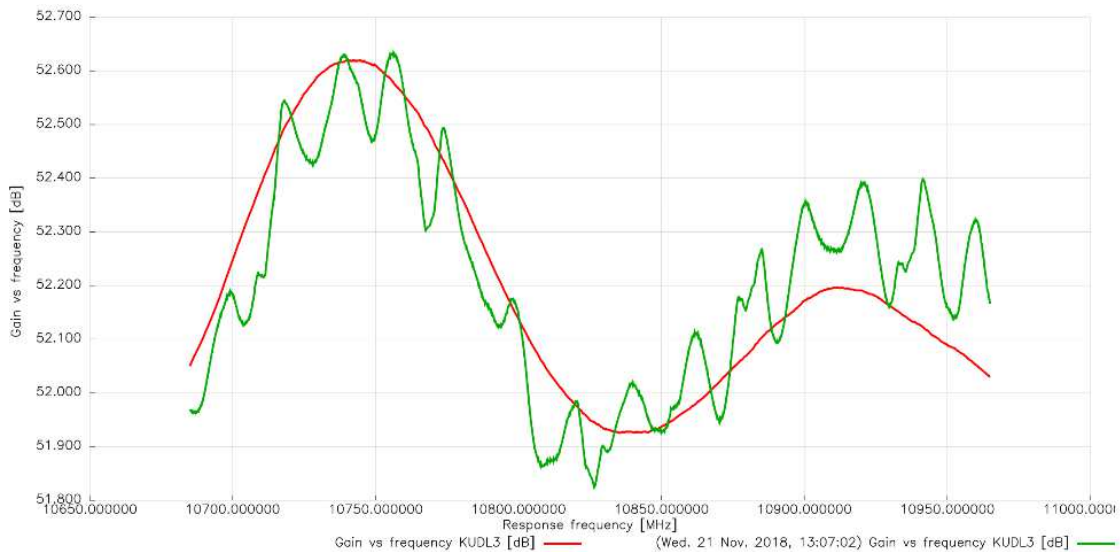


Figure 63: Multicarrier transmission gain measurements using internal loop calibration in the Ku-band.

It is almost impossible to skip the TAC calibrations and MCCs by executing a fast calibration with the same setup as MCMs.

We need a mixer in the internal loop to avoid the uncertainties of the ASA and AWG simultaneously. As can be seen in Figure 61, the TTC loopback (with a mixer) highlighted with green in the FrontEnd of the RF SCOE cannot be stimulated by the AWG. However, if there was a link between AWG and mixer input, the combination of the LO input and the AWG input produces intermodulation which corrupts the calibration measurements.

6.3.8.3 Calibration with theoretical chirp signal in the Ku-band

What does theoretical chirp mean?

The theoretical chirp is the ideal symmetric chirp which uses calcChirp procedure to generate IQ pattern in the time domain. By defining spacing, start and stop frequencies of the chirp spectrum the calcChirp procedure calculates the specific I/Q pattern. Then after applying FFT, the magnitude and phase of the chirp signal are extracted. The magnitude and phase of the theoretical chirp signal are calculated with a complex-valued FFT of length 4000 of the I/Q pattern of the chirp waveform. You can find the i-TCL implementation of the theoretical chirp in the attachment.

Because the chirp signal generated by AWG1 passes through the internal loop, it is required to calibrate the internal loop in order to take into account the path calibration factor for different frequencies. This is implemented via the procedure CAL_IQ.

There are two possibilities to achieve amplitude calibration of the chirp signal: Either one uses the Ka-band (27725 MHz in our case) or the Ku band (10825 MHz in our case) as the center frequency of the chirp signal. Both methods are tested, and it is more adequate to use the Ku Band as center frequency because also the ASA1 measures the amplitude response of the DUT in Ku Band.

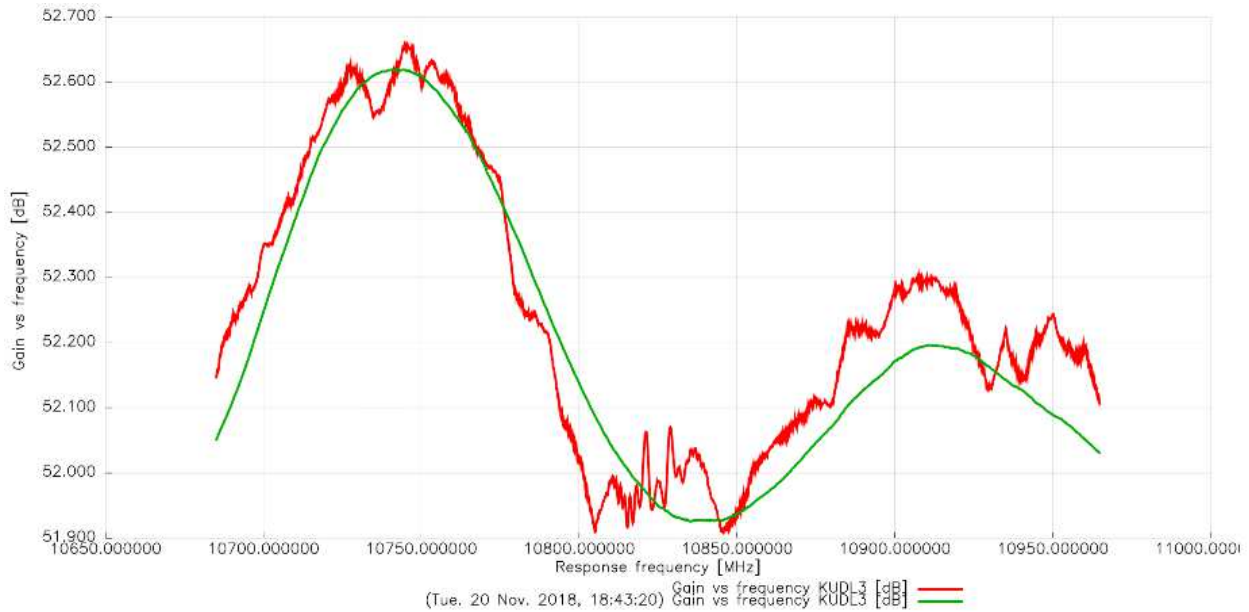


Figure 64: MC gain measurement uses theoretical chirp as calibration.

6.4 Impact of different chirp carrier spacings and input power levels on the group delay measurements

The group delay ripple is an important criterion to assess the quality of the measurement, the smaller the ripple, the more accurate the group delay measurement becomes. The MCM designed for OneWeb always uses 0.1 MHz spacing for the chirp signal for calibration and measurement. It would be interesting to measure the group delay for various different spacings and power levels of the chirp waveform, but the same measurement bandwidth.

The chirp signal with larger spacing has fewer carriers in the same bandwidth, therefore, the total power generated by AWG will be distributed over a smaller number of carriers. Therefore, it is expected that each carrier then has more power.

If the total power level of the chirp waveform is decreased, the difference in the group delay ripple for different spacings becomes stronger.

Figure 65 shows the group delay measurements for different spacings. The red, green, and blue curves show the group delay of the DUT with 0.5 MHz, 0.2 MHz, and 0.1 MHz spacing, respectively. When the total power level of the whole signal is somewhat decreased, as can be seen, the group delay of the DUT oscillates dramatically with an increase of spacing of the chirp waveform within the same measurement bandwidth.

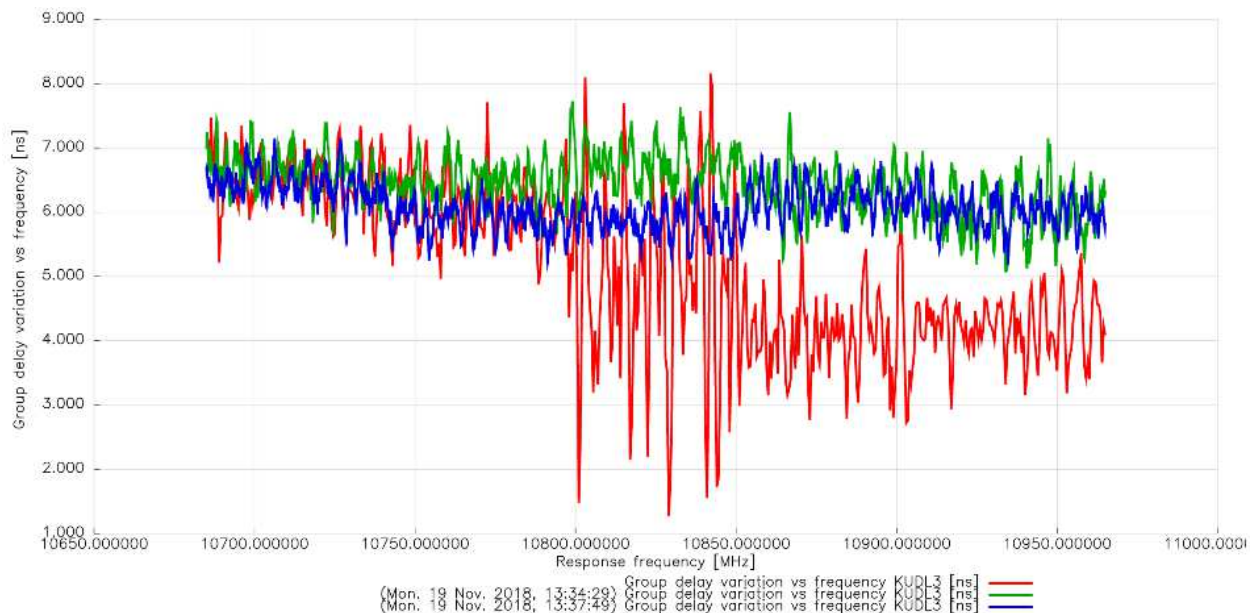


Figure 65: Absolute group delay measurements for different spacing with a low stimulus power level

Chapter 7

Transmission gain ripple compensation

Up to now all MCGD and gain measurements are executed with just a single Ka uplink path (KaUL3) and a single Ku downlink path (KuDL3). These two paths are selected for the best overall return loss by evaluating all return loss measurements for all paths. For Ka uplink paths, one out of four available paths is selected. For Ku downlink paths, one out of sixteen available paths is selected.

The concept for MCC is that the user executes MCC once with the selected paths. Afterward, the MCM designed for OneWeb is implemented with different combinations of the uplink and downlink paths by using MCC as a reference for the measurement. The MC gain measurements for different combinations of Ka and Ku paths is shown in Figure 66. The red curve (KaUL3-KuDL14) serves as our reference transmission gain trace for the MCM in the FWD link. This means that the MCC for the FWD link is executed with KaUL3 as an uplink path and KuDL14 as a downlink path, we call this path as reference calibration path.

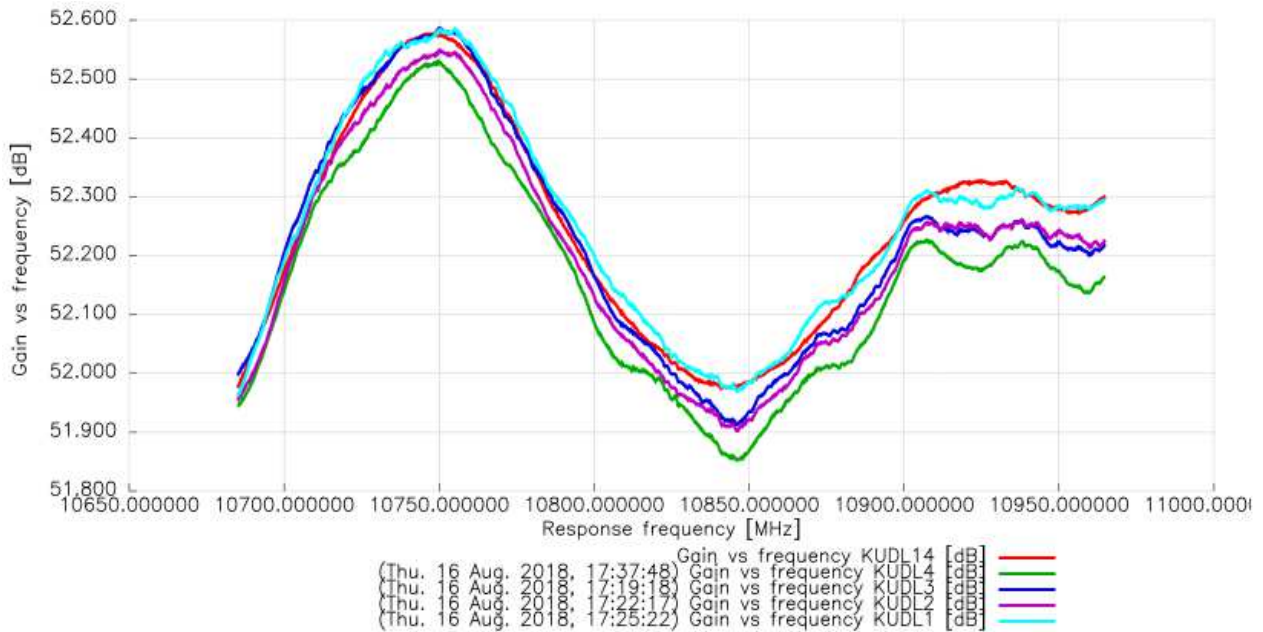


Figure 66: Gain curve for MCM with different paths.

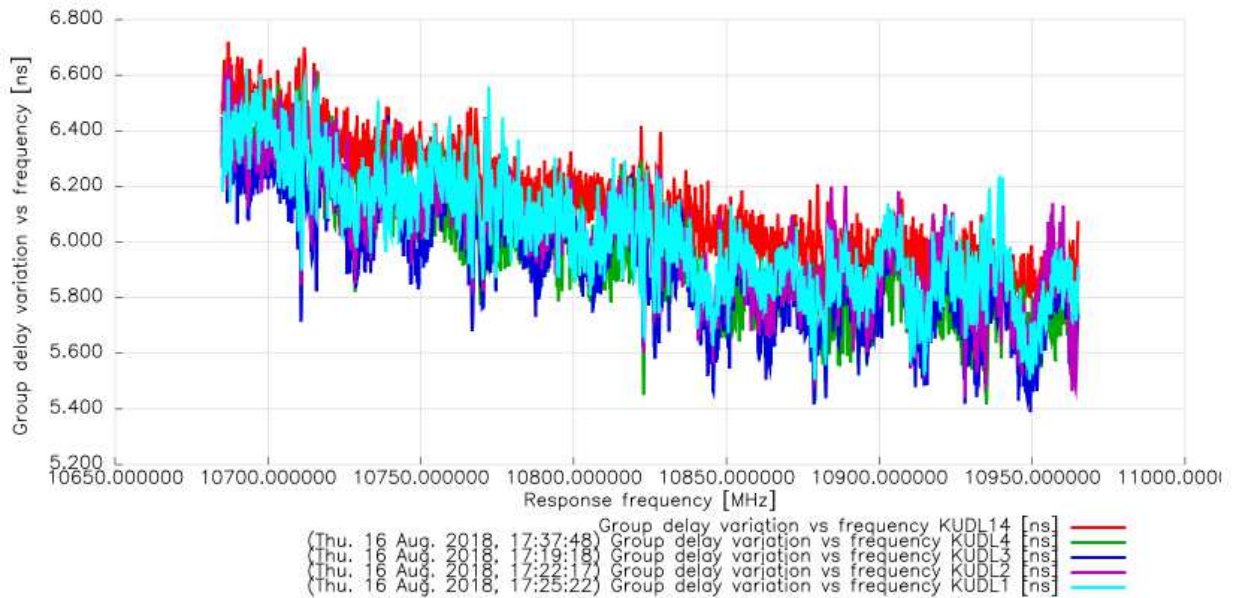


Figure 67: Group delay curve of MCM with various paths.

As can be seen in Figure 67, the group delay curve versus frequency for the reference path measurement looks smoother (with less ripple) than the other three. However, the others are also acceptable as a valid group delay for DUT.

If one runs the MCM for the FWD link with the same interface as used for calibration, then the gain curve of the DUT becomes flat without any ripple over frequency. In case, the user selects a different path (other than the one used for calibration), some ripple shows up in the measured gain curve. The reason for the ripples is impedance mismatches between the uplink harness cable's connector and the input connector of the DUT (UL) as well as mismatches between the downlink harness cable's connector and the output of the DUT(DL).

As explained, in MCMs when the calibration and measurements are executed with the same path, the difference between calibration and measurement does not depend on the impedance matching of the DUT and the harness cable's connector of the RF-SCOE. For power measuring devices such as a power meter or a signal analyzer, the calibration corrects for the mismatch loss of these devices. Once the MCM executed with a different path (other than the calibration path), the impedance matching of the interfaces used in MCC and MCM plays a significant role for the ripples which occur in MCMs.

A two-port BlackBox RF component terminated with a source and a load is shown in Figure 68. If we consider that a_1 is the incoming signal and b_1 is the outgoing signal at the reference plane of port 1. Equivalently, a_2 is the incoming signal and b_2 is the outgoing signal at the reference plane of port 2[16].

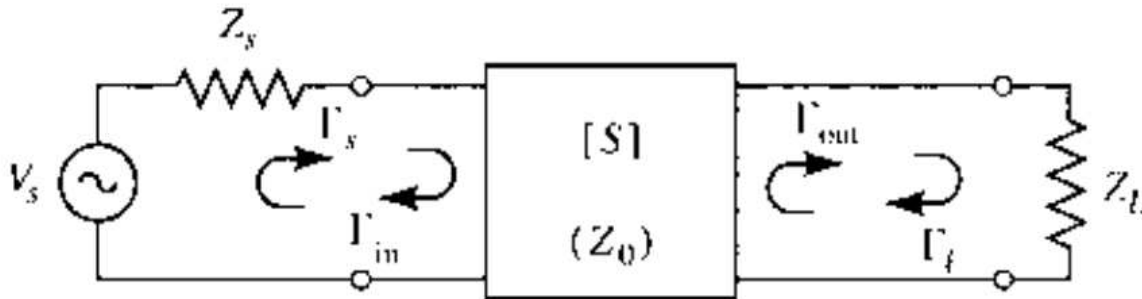


Figure 68: Two ports network connected to the source and load[16].

The input reflection coefficient of a two-port when port 2 is terminated with an arbitrary load.

$$\Gamma_{IN} = \frac{b_1}{a_1} = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1 - S_{22}\Gamma_L}$$

Equation (26): General input reflection coefficient.

The S-parameter measured at port 1 is assumed as:

$$S_{11} = \frac{b_1}{a_1} \text{ (when } a_2 = 0 \text{)}$$

Equation (27): Port 1 S-parameter.

The naming of the connectors at the VNA is somewhat confusing. When the user measures with port 1 of the VNA, then the naming of this connector seems to indicate that S_{11} of the DUT is being measured. However, the measurement result is ρ_{IN} of the DUT. Similarly, for port 2, it is not S_{22} of the DUT that is being measured, but ρ_{OUT} .

The VNA measures ρ_{11} instead of S_{11} . Therefore, the measurement results for ρ_{IN} is shown as the curve named “S11” in the VNA display.

Consider a signal generator with an output impedance Z_s , connected to a load Z_L , as shown in Figure 69. In uplink signal flow ρ_L is considered as reflection coefficient of the DUT input and ρ_g is considered as reflection coefficient at the harness cable connector.

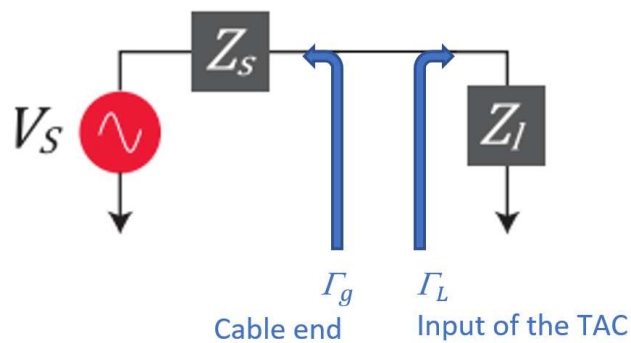


Figure 69: A generator connected to a load.

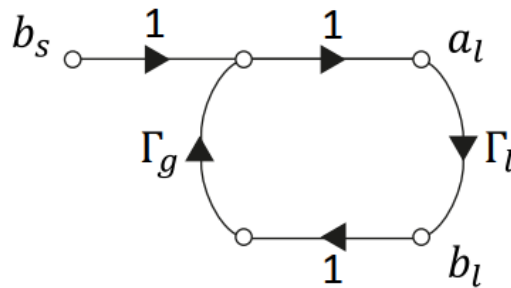


Figure 70: Signal flow graph for generator connected to load.

Relying on Mason’s non-touching loop rule for signal flow graphs

$$a_l = \frac{b_s}{1 - \Gamma_l \Gamma_g}$$

$$b_l = \frac{b_s \Gamma_l}{1 - \Gamma_l \Gamma_g}$$

Equation (28): Incident power and reflected power from the load.

The power absorbed by the load, P_a , is the difference between the power incident upon the load (a_l) and the power reflected by the load.

$$P_a = |a_l|^2 - |b_l|^2 = |b_s|^2 \frac{1 - |\Gamma_l|^2}{|1 - \Gamma_l \Gamma_g|^2}$$

Equation (29): Absorbed power by the load.

The term $1 - |\Gamma_l|^2$, Equation (29) represents mismatch loss related to the load which is solely a function of the load impedance. If the load impedance is not equal to the characteristic impedance, then this reduces the absorbed power relative to the power absorbed by a perfectly matched load.

Whereas the calibration of a power measuring device generally accounts for overall mismatch losses, accounting for the term $|1 - \Gamma_l \Gamma_g|^2$ represents the total mismatch loss due to multiple reflections between the load and the generator.

Total mismatch losses in decibels are calculated by

$$ML = 10 \log_{10}(|1 - \Gamma_l \Gamma_g|^2) = 20 \log_{10}(|1 - \Gamma_l \Gamma_g|)$$

Equation (30): Total mismatch losses in decibels[17].

The goal of this chapter is to eliminate the ripple caused by mismatches in MC gain measurement. To this aim, S-parameters need to be measured at the cable's connectors (all Ka uplink and Ku downlink, 1 to 4) and the TAC input/output. The MCC has to be executed once as the reference measurement by using the best interface in terms of return loss. This is KAUL3-KuDL3 for the available interfaces. The MCM must be executed for our reference interfaces and also for combinations of KaUL x -KuDL x , $x=1,2,4$.

The following factors potentially affect our MC gain measurements:

1. The R&S power sensor NRP33SN is used for uplink calibration. It has a return loss of almost 26 dB in the Ka-band frequency range. Further, the Rosenberger K-f-f adaptor between the harness cable and the PWS is needed. This adaptor has a very high return loss of more than 30 dB in the uplink frequency range. The insertion loss of this adaptor is set to be 0.06 dB (estimated average) in the Ka-band frequency range in our measurement routine. This value is determined and measured

with a network analyzer. The insertion loss of the adaptor is less important because the calibration has to be executed for all uplink cables with the same adaptor. This means that errors in the calibration are relative to each other and do not impact the ripple compensation error. The remaining uncertainty in the measurement is caused by the difference between the DUT input return loss (TAC input) and the power sensor.

2. A calibration coupler with a good return loss (<20 -dB) in the Ku band is used for the downlink calibration. This brings up the same issue of not measuring and calibrating with the same interface in our measurement.
3. The return loss measurement is executed with the E-CAL kit with one male port and one female port in order to measure the return loss of female (TAC-input) and male (harness cable end) connectors accurately.
4. The calibration is repeated shortly before the measurement to ensure that the temperature does not change between calibration and measurement.
5. After calibration, the harness cables should not be touched in order to keep the transmission characteristics of the cable constant. This is especially important for the phase stability of the measurement.

All interfaces with the original configuration from OneWeb have a very good return loss, higher than 20-dB. This means that the ripple does not vary significantly in comparison to the reference curve. However, some ripples could also be due to other factors that were mentioned above, and the compensation will not always work.

7.1 Transmission gain ripple compensation for uplink-side in the original setup

The MC transmission gain measurements are executed with the original configuration of the harness cable and DUT by keeping the downlink interface identical as in the calibration procedure. Only the uplink interfaces are changed and measured one after another. The transmission gain ripples in the measurement with different uplink interfaces are compensated up to a large extent as shown in Figure 71, Figure 72, and Figure 73.

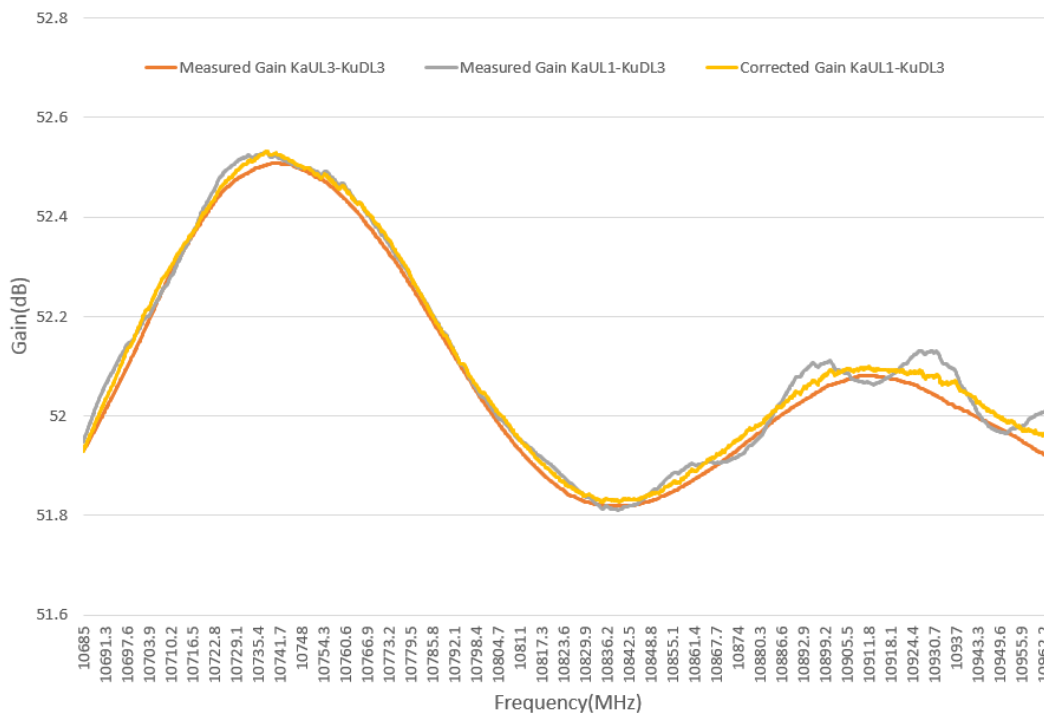


Figure 71: Ripple compensation uplink side for KaUL1-KuDL3.

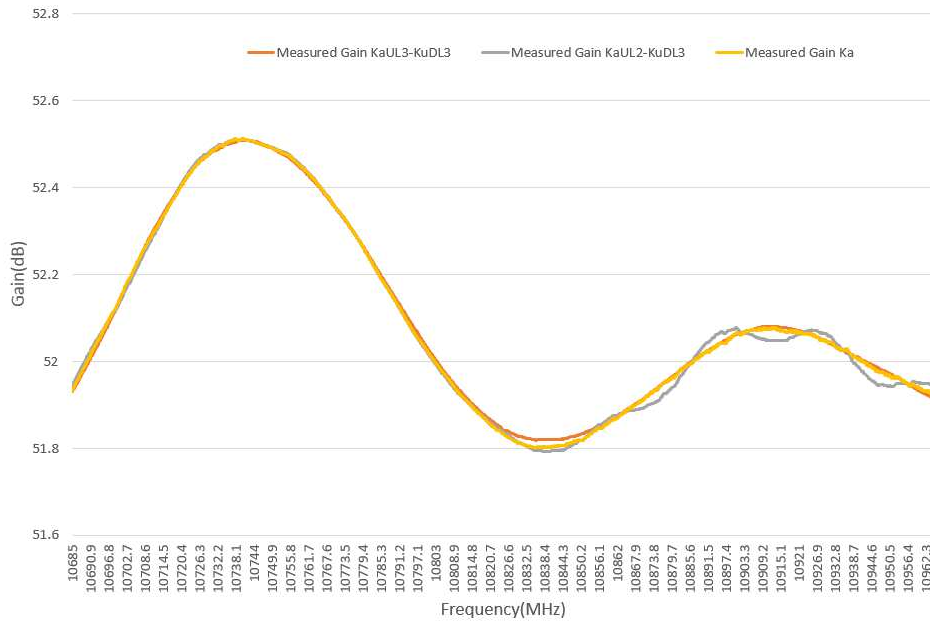


Figure 72: Ripple compensation uplink side for KaUL2-KuDL3.

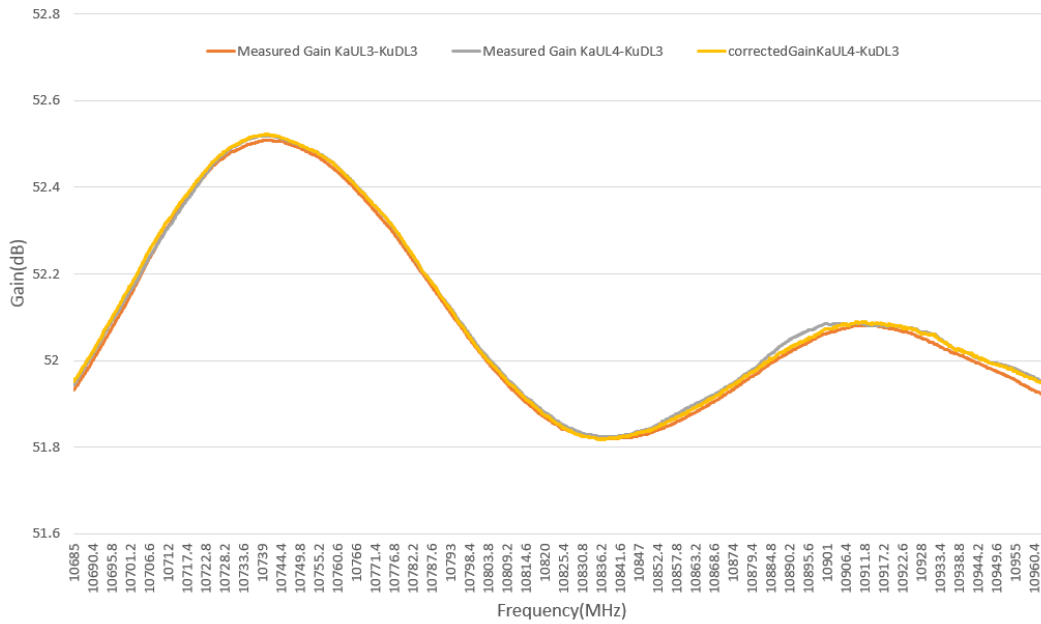


Figure 73: Ripple compensation uplink side for KaUL4-KuDL3.

7.2 Ripple compensation downlink-side return loss degradation

If the original setup of OneWeb is used, then changing only the downlink harness cables and keeping the uplink cables identical to the calibration routine will lead to an unacceptable result after applying the ripple compensation. The downlink interfaces are more prone to errors in the ripple compensation due to the difference between the return loss of the calibration coupler connector (for downlink calibration) and the return loss of the PWS (for uplink calibration). The power sensor has better return loss and the calibration is more accurate than the downlink calibration where the return loss of the coupler plays an important role in calibration accuracy.

For the purpose of illustration of the ripple compensation procedure in the downlink path, we deliberately degrade the return loss of the downlink interfaces by inserting an additional low-quality RF component (e.g. a cable, adaptor, or attenuator).

The return loss of the TAC output is degraded by concatenating a hybrid coupler to the TAC output. One of the unused ports of the hybrid coupler is terminated with a 3-dB attenuator and the other port is left unterminated.

The return loss of the downlink cable is degraded by adding a selected additional SMA cable and an f-f adaptor. In Figure 74, Figure 75 and Figure 76, firstly, an MCC is performed with KaUL3 from uplink cables and KuDL3 from downlink cables. Secondly, an MCM is performed by keeping the uplink cable (KaUL3) unchanged. three MCMs are repeated for various downlink cables (KuDL1, KuDL2, and KuDL4). The result illustrates that the measured gain trace in grey color is improved to a large extent. The corrected gain trace is shown in yellow color.

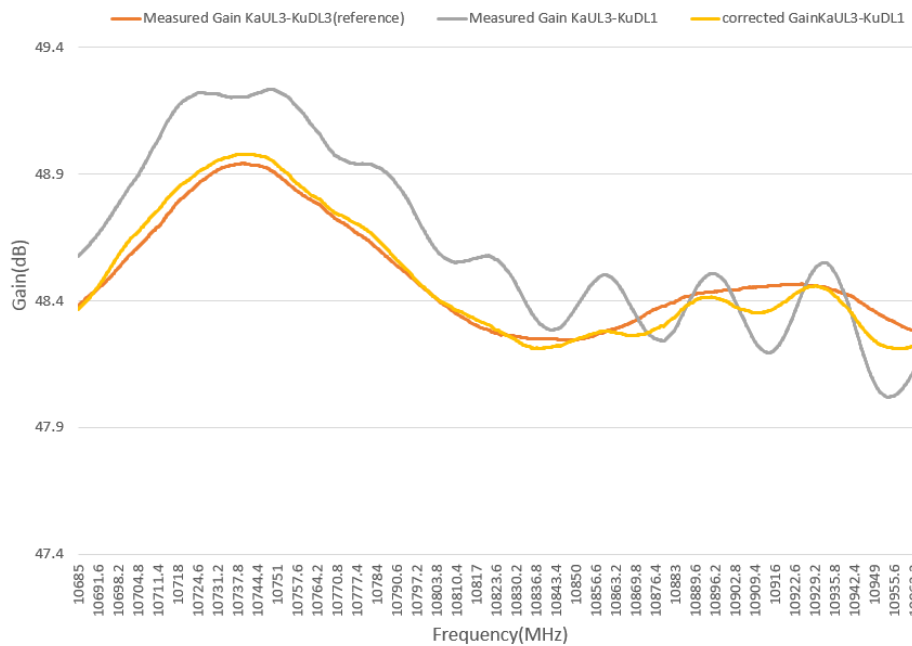


Figure 74: Ripple compensation downlink side for KaUL3-KuDL1.

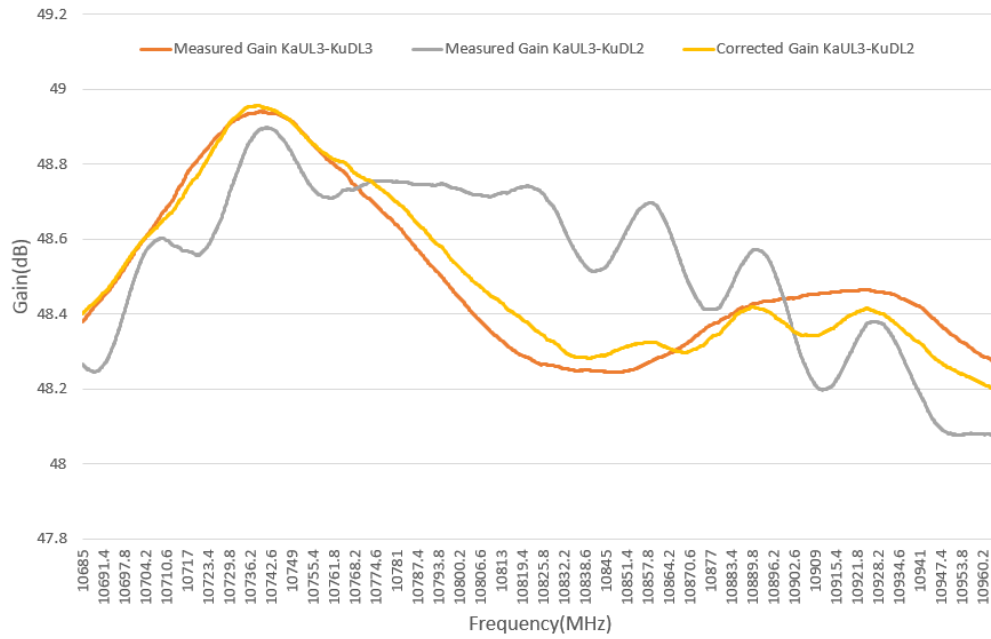


Figure 75: Ripple compensation downlink side for KaUL1-KuDL2.

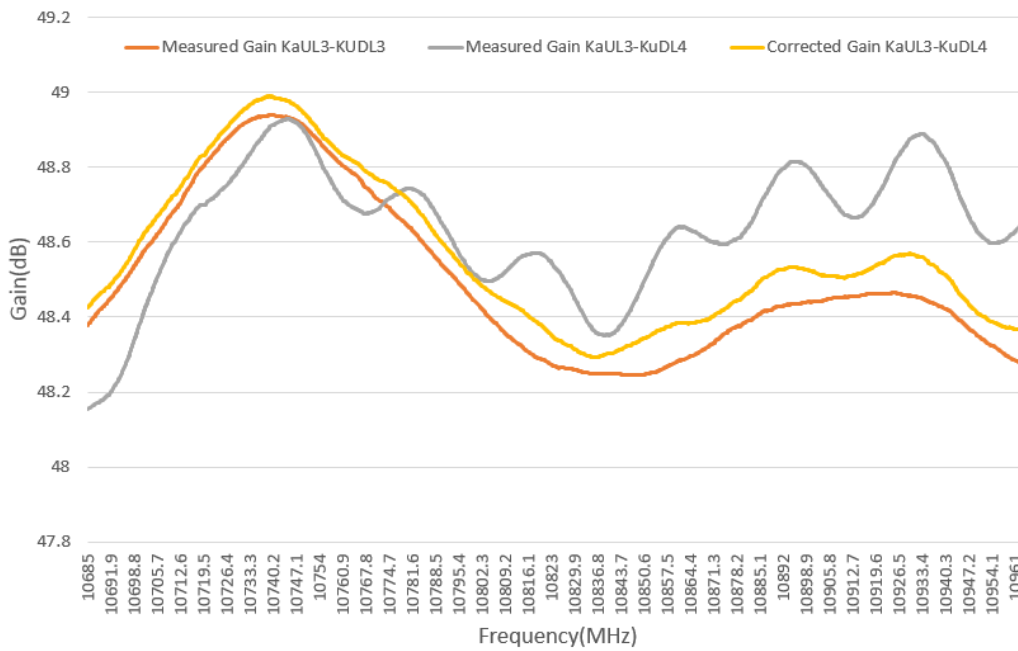


Figure 76: Ripple compensation downlink side for KaUL1-KuDL4.

7.3 Ripple compensation two-sided by return loss degradation

The MC gain measurement needs to be repeated for all combinations of KaUL and KuDL interfaces. This means that the forward gain measurement is executed for all reference interfaces. Both uplink and downlink cables are exchanged at each measurement step. In order to calculate mismatch losses for both uplink and downlink connection, the return loss of all uplink and downlink cables, TAC input and TAC output is measured and taken into account in the ripple compensation. In order to highlight the gain differences between the reference and normal cables, the return loss of all interfaces is degraded to some extent but not for the reference interfaces. The degradation procedure for the return loss of the DUT and harness cable is implemented with a 3-dB terminated hybrid coupler for the DUT (on both sides) and by attaching an f-f adaptor plus cable.

In

Figure 77, Figure 78, and Figure 79, the reference measurements are performed with the same interface as calibration (KaUL3-KuDL3). The reference measurements are shown as the orange-colored traces. The measurements with the other combination of interfaces are shown as the grey colored traces. The corrected gain trace, measured by other combinations of the interfaces are shown as yellow-colored traces.

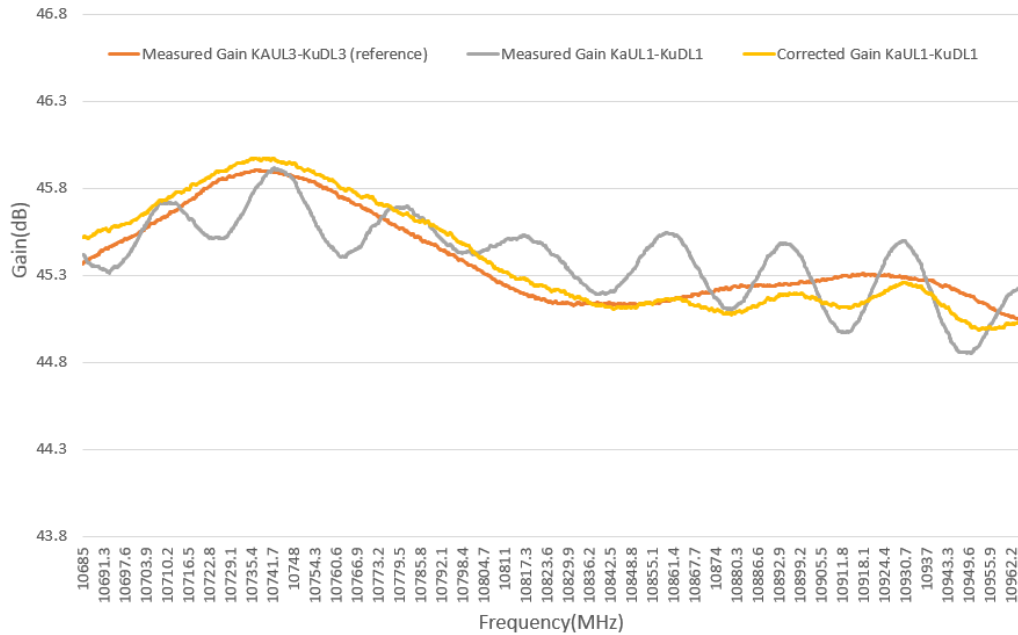


Figure 77: Ripple compensation two-sided for KaUL1-KuDL1.

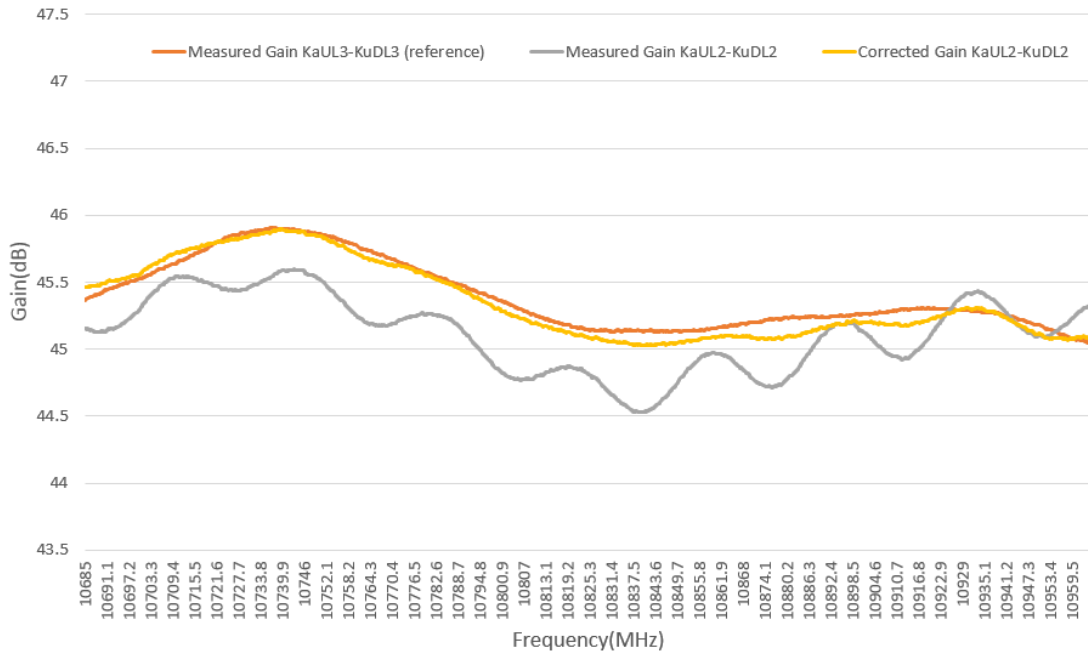


Figure 78: Ripple compensation two-sided for KaUL2-KuDL2.

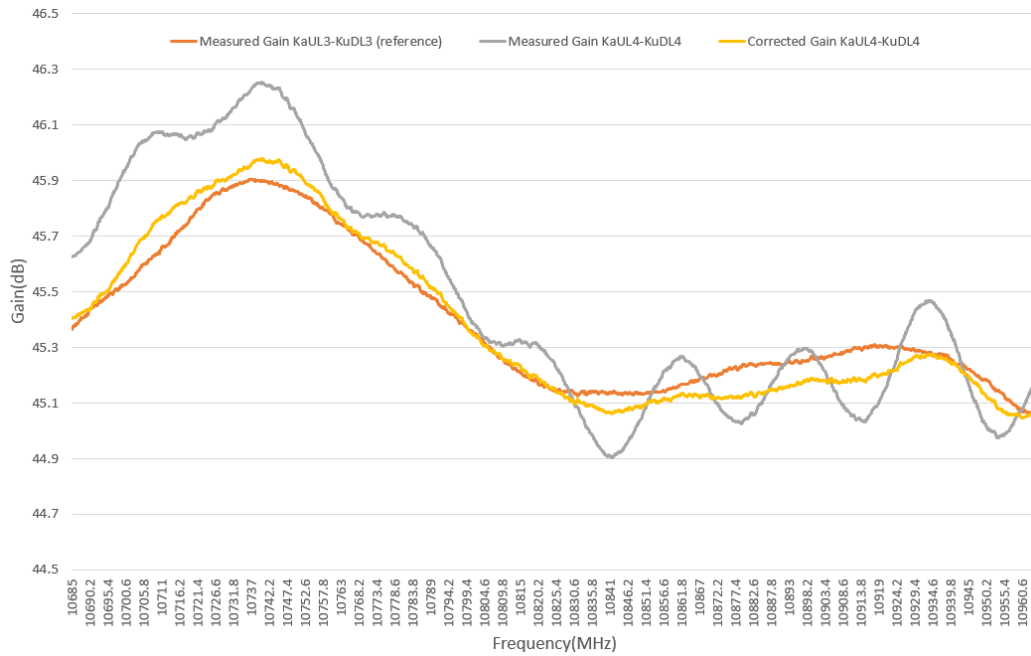


Figure 79: Ripple compensation two-sided for KaUL4-KuDL4.

Chapter 8

Conclusions and future work

In satellite communication, the operator inevitably has always the difficulty that the satellite is not accessible after it's launched to space. This thesis implements the transmission gain and (absolute and relative) group delay measurements on a broadband satellite with has 280 MHz operation bandwidth. The satellite performs frequency conversion in the Ka-Ku frequency bands and the LO of the satellite is not accessible during calibration and measurement. This thesis illustrates the possibility of the transmission gain and absolute group delay measurements by running specific calibrations (turn around convertor and multicarrier calibrations) prior to the measurement. The thesis proves that the absolute group delay measurements are possible by performing a vector network analyzer group delay measurement of the satellite prior to the launch of the satellite to the orbit.

The bandwidth of the measurement is also a determining factor in the difficulty of the measurement. Due to the large operation bandwidth of the satellite and limitation in our instrument's capability, the group delay of the satellite cannot be measured by broadband phase acquisition of the signal. Therefore, the bandwidth of the measurement is divided into smaller sub-bands. The division of measurement bandwidth brings unwanted spikes in the transmission gain and group delay measurement traces. Finally, all these undesired errors have been corrected by implementing interpolation and removing outliers from transmission gain and absolute group delay measurements.

The Rohde & Schwarz K17 firmware option cannot be utilized for the calibration and measurement of the multicarrier group delay with FTD. Hence, this thesis offers a method to measure the amplitude and phase of any kind of input signal (including MC chirp signal) from the time domain I/Q pairs. The amplitude error

of the measurement's signal in the carrier by carrier comparison with the K17 option of the Analog Spectrum Analyser(ASA) is always less than 0.08 dB. Subsequently, the group delay over the bandwidth of the measurement driven from the difference of the calculated phase in the measurement and calibration. The measured absolute group delay has a difference of 0.22 ns in comparison to the provided amplitudes and phases by the K17 option of the ASA.

The thesis presents an approach to skip the annoying, repetitive, and time-consuming turn around convertor calibrations and MultiCarrier Calibrations(MMCs) for the transmission gain measurements. The whole calibration part is summarized in an internal calibration to save time in the calibration part of the measurement. It has been proved that the result of the transmission gain measurement using an internal calibration deviates to a maximal extent of 0.2 dB compared to the measured transmission gain using the time-consuming turn around convertor and MCCs procedures. This means, there is still a chance for future work to discover new methods or improve the existing methods for more accurate transmission gain measurements. However MCC is executed once for two specific up/downlink paths, the radio frequency payload test system Front End(FE) has the possibility to measure the transmission gain of the satellite with different paths. The transmission gain measurements of the satellite with non-calibrated paths result in slightly different transmission gain traces compared to the measurement from calibrated paths. Therefore, a solution for obtaining similar transmission gain traces is provided in case of executing the gain measurement of the satellite with non-calibrated up/downlink FE paths. The solution retrieves the correct gain curve from a faulty trace by an error of less than 0.2 dB in the worst-case scenario. The possible causes of the error which left uncorrected at the measurements are discussed. Therefore, there are still open points for the future to improve the error in the frequency translating device transmission gain measurements.

Bibliography

- [1] *Novel Approaches for Measuring Frequency Converted Group Delay*. 2010.
- [2] *Understanding the Fundamental Principles of Vector Network Analysis*. Application Note, 2012.
- [3] Helwett Packard Co., *Improving Network Analyzer Measurements of Frequency-translating Devices*. Application Note 1287-7, 1998.
- [4] M. K. and J. D. My Le Truong, “Highly Accurate Absolute Group Delay Measurement Technique for Frequency Translation Devices,” p. 6, 1998.
- [5] Hewlett-Packard, “User’s Guide HP 8753D Network Analyzer,” p. 678, 1997.
- [6] *R&S®FSW-K17 Reference Manual*, Version 22. 2018.
- [7] *R&S®FSW I/Q Analyzer and I/Q Input Interfaces User Manual*, Version 31. 2018.
- [8] SIEMENS AG, *Fundamentals of DSP*. 2017.
- [9] S. W. smith, *The Scientist and Engineer’s guide to Digital Signal Processing*, Second Edi. San Diego, California, 1999.
- [10] “TCL Manual Page.” [Online]. Available: <https://www.tcl.tk/man/>.
- [11] *RF SCOE Technical Specification*, Version 3. 2016.
- [12] A. J. Hermann Danzer, Marco Conte, *PTS-TCR User Manual*, Version 10. 2018.
- [13] *PNA Network Analyser Help Tutorial*. 2012.
- [14] “Group Delay measurements with signal and spectrum analysers,” p. 17.
- [15] *R&S®FSW Signal and Spectrum Analyzer User Manual*, Version 36. 2018.
- [16] David. M Pozar, *Microwave Engineering*, Fourth Edi., vol. 8, no. 2. John Wiley & Sons, Inc., 1989.
- [17] “Using Gamma, S-Parameter Correction, and Real-Time Measurement Uncertainty (RTMU) to Improve Measurement Accuracy,” p. 15, 2014.

Appendix I

```

1. proc ASA1.getdata_mcsignals_iq {args} {
2.   Log d "ASA1.getdata_mcsignals2: $args" ;# Version with reading IQ data in IQ analys
   er mode without K17
3.   #%ProcedureRange
4.   set range(-center)      {-minmax 0:+ -values {}}
5.   set range(-span)       {-minmax 0:+}
6.   set range(-spacing)    {-minmax 0:+}
7.   set range(-att)        {-minmax 0:+}
8.   set range(-refLevel)   {-minmax -:+}
9.   set range(-preAmp)     {-values {} on off y n yes no t f true false 1 0} -
   case no}
10.  set range(-avgcnt)     {-minmax 1:+}
11.  set range(-traceAver)  {-values {} on off y n yes no t f true false 1 0} -
   case no}
12.  set range(-trigger)    {-values {IMM EXT EXT2 EXT3}}
13.  set range(-sweep_time) {-minmax -1:+ -values {}}
14.  set range(-offset)     {-values {EST FIX}}
15.  #%ProcedureArgs
16.
17.  set options(-center)    {}
18.  set options(-span)      100
19.  set options(-spacing)   0.1
20.  set options(-att)       10
21.  set options(-refLevel)  0
22.  set options(-preAmp)    {}
23.  set options(-avgcnt)    1
24.  set options(-traceAver) 0
25.  set options(-trigger)   {EXT}
26.  set options(-sweep_time) {}
27.  set options(-offset)    {FIX}
27.  #%EndArgs
28.  set allowed [array names options]
29.  array set options $args
30.  ## check options
31.  checkCompAllowed options $allowed
32.  checkCompRange options range
33.
34.  if {[valbool [ASA1 cget -simulated]]} {
35.    set pfd $::SPOOL_PATH
36.    if {[string range $pfd 3 11] == "Documents"} {set pfd "D:/temp/"}
37.    if {[string range $pfd 0 7] == "C:/Users"} {set pfd "D:/temp/"}
38.    set fileName [file join $pfd "mcreresults_ref.xls"]
39.
40.    if {[catch {set filep [open $fileName r]}]} {
41.      set number [expr {1 + int($options(-span)/$options(-spacing))}]
42.      set freq_list {}
43.      set power_list {}
44.      set phase_list {}
45.      set powerOffset [expr {rand() - 0.5}]

```

```

46.         set phaseOffset [expr {180. * (rand() - 0.5)}]
47.         set powerSlope [expr {0.01 * (rand() - 0.5)}]
48.         set phaseSlope [expr {1. * (rand() - 0.5)}]
49.         if {$options(-center) < 26000} {set period 3.456} else {set period 5.678}
50.         for {set x 0} {$x < $number} {incr x} {
51.             set freq [expr {round(100000. * ($options(-
center) + int($x - $number/2) * $options(-spacing)))/100000.}]
52.             lappend freq_list $freq
53.             lappend power_list [expr {$powerOffset + $powerSlope * $x + rand() - 0.
5 + 1.1 * sin(6.28 * $freq / $period)}]
54.             lappend phase_list [normalizedPhase [expr {$phaseOffset + $phaseSlope *
$x + 0.0002 * (rand() - 0.5) + 0.00021 * sin(6.28 * $freq / $period)}]]
55.         }
56.     } else {
57.         Log h "ASA1.getdata_mcsignals1: read from file $fileName !"
58.         # ASA1.getdata_mcsignals1 -center 27200 -span 159 -spacing 0.5
59.         if {$options(-
center) > 26767} {set amps 1; set phas 3; set offs 1415} else {set amps 4; set phas 6;
set offs 0}
60.         set fstart [expr {$options(-center) - $offs - $options(-span) / 2.}]
61.         set fstop [expr {$options(-center) - $offs + $options(-span) / 2.}]
62.         Log d "ASA1.getdata_mcsignals1: fstart=$fstart fstop=$fstop offs=$offs"
63.         set freq_list {}; set phase_list {}; set power_list {}
64.         while {[eof $filep]} {
65.             set tmp [string trim [gets $filep]]; check_events
66.             if {[isnumeric [lindex $tmp 0]]} {
67.                 if {([lindex $tmp 0] - $fstart) > -
0.01 && ([lindex $tmp 0] - $fstop) < 0.01} {
68.                     if {$amps == 1} {
69.                         lappend freq_list [format %.1f [expr {[lindex $tmp 0] + $of
fs}]]
70.                     } else {
71.                         lappend freq_list [lindex $tmp 0]
72.                     }
73.                     scan [lindex $tmp $amps] {%e} pw
74.                     lappend power_list $pw
75.                     lappend phase_list [lindex $tmp $phas]
76.                 }
77.                 if {abs([lindex $tmp 0] - $fstop) < 0.05} {break}
78.             }
79.         }
80.         close $filep
81.         ## cut off overlapping ranges
82.         for {set istart 0} {$istart < [llength $freq_list] / 2.} {incr istart} {
83.             if {([lindex $freq_list $istart] - [lindex $freq_list $istart+1]) > 0}
{break}
84.         }
85.         incr istart
86.         if {$istart > [llength $freq_list] / 3.} {set istart 0}
87.         for {set istop [expr {[llength $freq_list] - 1}]} {$istop > [llength $freq_
list] / 2.} {incr istop -1} {
88.             if {([lindex $freq_list $istop] - [lindex $freq_list $istop-
1]) < 0} {break}
89.         }
90.         incr istop -1
91.         if {$istop < [llength $freq_list] / 1.5} {set istop [llength $freq_list]; i
ncre istop -1}
92.         Log d "ASA1.getdata_mcsignals1: istart=$istart istop=$istop "

```



```

93.         set freq_list [lrange $freq_list $istart $istop]
94.         set power_list [lrange $power_list $istart $istop]
95.         set phase_list [lrange $phase_list $istart $istop]
96.     }
97.     ## build up result list
98.     set result [list -frequency $freq_list -phase $phase_list -power $power_list]
99.     Log d "ASA1.getdata_mcsignals1: simulated freq_list [lindex $freq_list 0] [lind
ex $freq_list end] values=[llength $freq_list]"
100.        return $result
101.    } else {
102.        ##### not simulated #####
103.        set sampleRateUncorr [expr {$options(-span)/ 0.8}]
104.        set recordLength1Uncorr [expr {$sampleRateUncorr / $options(-
spacing)}]
105.        set corr [expr {$options(-
spacing) * (1. - ($recordLength1Uncorr - int($recordLength1Uncorr))}]
106.        set sampleRate [format %.3f [expr {$sampleRateUncorr + $corr}]]
107.        set recordLength1 [expr {int(round($sampleRate / $options(-
spacing))}]
108.        set nrOfPeriods [expr {int(3700000 / $recordLength1)}]
109.        set recordLength [expr {$recordLength1 * $nrOfPeriods}]
110.        Log h " recordLength = $recordLength options(-spacing)=$options(-
spacing) sampleRateUncorr=$sampleRateUncorr recordLength1Uncorr=$recordLength1Uncorr c
orr=$corr recordLength1=$recordLength1 recordLength=$recordLength nrOfPeriods=$nrOfPe
riods sampleRate=$sampleRate"
111.
112.        ASA1 preset
113.        ASA1 write "INST:SEL IQ"
114.        ASA1 write "INIT:CONT OFF"
115.        ASA1 write "TRAC:IQ:DATA:FORM IQP"
116.        ASA1 setval center $options(-center) att $options(-
att) ref_level $options(-refLevel)
117.
118.        ASA1 write_read "TRAC:IQ:SET NORM,0,$sampleRate MHz,EXT,POS,0,$recordLen
gth; *OPC?"
119.        ##ASA1 write_read "SWE:TIME $measTime; *OPC?"
120.        if {$options(-
span) < 64} {ASA1 write "TRAC:IQ:WBAN:MBW 80MHZ"} else {ASA1 write "TRAC:IQ:WBAN:MBW MA
X"}
121.        idle 100
122.        ASA1 write_read "INIT:IMM;*OPC?" -read_timeout 90000 ;# sweep once
123.
124.        if {$options(-span) < 64} {
125.            set triggerOffset [ASA1 write_read "TRAC:IQ:TPIS?"]
126.            Log h "triggerOffset=$triggerOffset seconds or [format %.3f [expr $t
riggerOffset*1e9]]ns"
127.        }
128.
129.
130.        set startTime [getclock -float]
131.        ASA1 write "FORM REAL,32"
132.        set res [[ASA1 cget -
connIfObj] write_read_scpi_block "TRAC:IQ:DATA:MEM?" -read_mode sync -
read_timeout 60000]
133.        set startTimeX [getclock -float]
134.        binary scan $res f* res

```

```

135.      Log h "Reading and converting in Binary mode takes [expr {[getclock -
float] - $startTime}]"
136.      Log h "converting takes [expr {[getclock -float] - $startTimeX}]"
137.
138.      set startTime1 [getclock -float]
139.      set magList {}; set phaseList {}; set freqList {}
140.      ##### doFFT #####
141.
142.      set startTime [getclock -float]
143.      set tmpPlan [fftw::plan [expr {[llength $res]/2}] 1]
144.      fftw::clear $tmpPlan
145.      set ind 0
146.      foreach {i q} $res {fftw::setAt $tmpPlan 0 $ind [list $i $q]; incr ind}

147.      fftw::execute $tmpPlan 0
148.      set fft [fftw::get $tmpPlan 0]
149.      fftw::destroy $tmpPlan
150.      unset res
151.
152.      ##### calculate Phase and Mag #####
##
153.      set startTime [getclock -float]
154.
155.      set lenfft [llength $fft]
156.      set factorPi [expr {45/atan(1)}]
157.      set scaleFactor [expr {20./pow($lenfft, 2)}]
158.      set center [expr {[ASA1 write_read "FREQ:CENT?"] / 1e6}]
159.
160.      for {set i $nrOfPeriods} {$i < [expr {$lenfft / 2}]} {incr i [expr {$nr0
fPeriods}]} {
161.          set freq [expr {$center - ($i/$nrOfPeriods)*$options(-spacing)} ]
162.          if {$freq < [expr {$center - $options(-span)/2.}] } {break}
163.          lappend freqList $freq
164.          lassign [lindex $fft [expr {$lenfft - $i}]] e1 e2
165.          lappend magList [expr {10*log10($scaleFactor*($e1*$e1 + $e2*$e2))}]

166.          lappend phaseList [expr {atan2($e2,$e1)*$factorPi}]
167.      }
168.
169.      set magList [lreverse $magList]
170.      set phaseList [lreverse $phaseList]
171.      set freqList [lreverse $freqList]
172.      for {set i 0} {$i < [expr {$lenfft / 2}]} {incr i [expr {$nrOfPeriods}]}
{
173.          set freq [expr {$center + $i/$nrOfPeriods*$options(-spacing)} ]
174.          if {$freq > [expr {$center + $options(-span)/2.}] } {break}
175.          lappend freqList $freq
176.          lassign [lindex $fft $i] e1 e2
177.          lappend magList [expr {10*log10($scaleFactor*($e1*$e1 + $e2*$e2))}]

178.          lappend phaseList [expr {atan2($e2,$e1)*$factorPi}]
179.      }
180.
181.      if {$options(-span) < 64} {
182.          set slopeCorr [expr {$triggerOffset*$options(-spacing)*1e6*360}]
183.          set x 0
184.          foreach ph $phaseList {

```

```
185.             lset phaseList $x [normalizedPhase [expr {$ph - $x*$slopeCorr}]]
186.             incr x
187.         }
188.     }
189.
190.     Log h "mag and phase calculation takes [expr {[getclock -
float] - $startTime}]"
191.     Log h "FFT method takes in total [expr {[getclock -
float] - $startTime1}]"
192.
193.     set filep [open /home/scoe-usr/Desktop/fft.csv w]
194.     foreach freq $freqList amp $magList ph $phaseList {puts $filep "$freq\t
$amp\t$ph"}
195.     close $filep
196.
197.     set result [list -frequency $freqList -phase $phaseList -
power $magList]
198.     return $result
199. } ;# end of else "simulated"
200. }
```

Appendix II

```

1.  ## source $TESTSEQ_PATH/MEAS_MCARRIER.itcl
2.
3.  body ONEWEB-RFTS_CCC_MEAS_MCARRIER::execute {args} {
4.      ## get cc values
5.      array set options [getOption $args -check_result]
6.
7.      ## make all upper case
8.      foreach option [array names options] {set options($option) [string toupper $options
9.      ($option)]}
10.     ## options: respPath, respCenter, span, carrierSpacing, aperture, stimPath, stimCen
11.     ter, stimPower,
12.     ##          useCalSet, calAsaPwmResp, avgCnt, atten, refLevel, resultCount, evalCen
13.     ter, evalSpan
14.
15.     ## less parameters for OneWeb:
16.     set options(preAmp) 0
17.     ## trace averaging currently not working due to bug in K17
18.     set traceAver 1
19.
20.     ## log
21.     set sdesc "$loggingTitle: Initialize ..."
22.     set ::mainHelpvar $sdesc
23.     set seqTime [getclock -float]
24.     Log n $sdesc
25.
26.     set dbgf [valbool [CONF getval enableCsvResults]];## dbgf = true -
27.     > write to file
28.
29.     if {$options(evalCenter) == "AUTO"} {set options(evalCenter) $options(respCenter)}
30.
31.     if {[string is double $options(evalCenter)] && [string is double $options(evalSpan)
32.     ]} {
33.         set evaluate 1
34.         if {abs($options(evalCenter) - $options(respCenter)) > $options(span) / 2.} {er
35.         ror "evalCenter outside span!"}
36.     } else {
37.         set evaluate 0
38.     }
39.
40.     ## clear old results
41.     eval setDefault [getcompOption -idlist result]
42.
43.     ## switchUplink returns the pathname and attaches external cables ("Aext") to the s
44.     ignal path
45.     set ulPath [switchUplink -path $options(stimPath) -source "AWG1"]
46.     set dlPathSa [switchDownlink -path $options(respPath) -instrument "SA"]
47.     set dlPathPm [switchDownlink -path $options(respPath) -instrument "PM"]
48.
49.     ## splitSpan creates center lists and span lists with span =maxSpan if necessary
50.     set spacing [CONF getval mcCarrierSpacing]
51.     set result [splitSpan -center $options(stimCenter) -span $options(span) -
52.     spacing [expr {round(1e6 * $spacing) / 1.e6}]]
53.     set stimCenterList [getOption $result -centerList]
54.     set spanList [getOption $result -spanList]

```

```

45.   Log d "$loggingTitle: stimCenterList = $stimCenterList ... spanList = $spanList ..
46.   ."
47.   ## check span/spacing
48.   if {[lmax $spanList] / [expr {round(1e6 * $spacing) / 1.e6}] > 160001} {error "span/spacing must be < 160001!"}
49.
50.   set respFreqs {}
51.   set ampDiffs {}
52.   set phaseDiffs {}
53.   set respStop 0
54.   ASA1 preset
55.   PWS1 preset
56.   AWG1 preset
57.
58.   if {[valbool $options(useCalSet)]} {
59.     set fbid [fb new [expr {3 + 2 * [llength $stimCenterList]}]]
60.   } else {
61.     set fbid [fb new [expr {3 + 3 * [llength $stimCenterList]}]]
62.   }
63.   check_events
64.
65.   Log t "$loggingTitle: measure frequency offset"
66.   ## setup CW signal with levelling, set all uplinks paths to off position
67.   set tcrLink 0
68.   if {[regexp "TCR" [getScoeUUID]]} {
69.     if {[valbool [UCV1 getval state]]} {
70.       set sc16pos [SC16 getval position]
71.       if {$sc16pos == 5} {
72.         set tcrLink 35
73.       } elseif {$sc16pos == 6} {
74.         set tcrLink 37
75.       }
76.     }
77.   }
78.   for {set i 19} {$i < 39} {incr i} {
79.     if {$i != $tcrLink} {SC$i setval position 2}
80.   }
81.   SC02 setval position 2
82.
83.
84.   ## measure gain at two frequencies calGainFreq and calGainFreq2 for tilt correction
85.   set calGainFreq [format %.1f [expr {$options(stimCenter) - $options(span) / 4.}]]
86.   set calGainRespFreq [expr {$calGainFreq + $options(respCenter) - $options(stimCenter)}]
87.
88.   set pathFactor [$ulPath calcPathFactor -center $calGainFreq]
89.   AWG1 setval center $calGainFreq state "ON"
90.   set difference [setMCPower -path $ulPath -center $calGainFreq -ulPower $options(stimPower) -pathFactor $pathFactor]
91.
92.   ## switch signal to uplink if target power is achieved
93.   if {abs($difference) <= 0.1} {
94.     $ulPath setval state on
95.   } else {
96.     if {[valbool $::SCOE_CONFIG(startup.Simulated)]} {

```

```

97.         error "Target power $options(stimPower) +/-
0.5 dBm could not be set! difference=$difference dB"
98.     } else {
99.         Log w "$loggingTitle: Simulated target power $options(stimPower) +/-
0.1 dBm could not be set! difference=$difference dB"
100.    }
101.    }
102.    $dlPathSa setval state "ON"
103.
104.    ## measure respPower and frequency offset with ASA
105.    ASA1 setval center $calGainRespFreq
106.    set asaSettings [calcAsaRefAtt -att $options(atten) -path $dlPathSa -
center $calGainRespFreq -preamp $options(preAmp)\
107.        -reflevel $options(refLevel) -steps "CONF"]
108.    set preAmp [getOption $asaSettings preamp_mode]
109.    set refLevel0 [getOption $asaSettings ref_level]
110.    set att0 [getOption $asaSettings att]
111.    set attGain [max 5 $att0]
112.
113.    ASA1 setval inst_mode "SAN" center $calGainRespFreq att $attGain ref_level $
refLevel0 span 0.1\
114.        rbw 0.002 sweep_time 50 sweep_mode "SINGLE" ;#yig_filter "OFF"
115.
116.    ASA1 marker -action {peak counter} -resolution 0.0000001 -sweep yes
117.    set measurement [ASA1 getdata_marker -sweep yes]
118.    set carrierFreq [getOption $measurement -frequency]
119.    set respOffset [expr {$carrierFreq - $calGainRespFreq}]
120.    set powerAsaRespRaw [getOption $measurement -power]
121.
122.    if {[ASA1 isSimulated]} {set respOffset -
0.001; set powerAsaRespRaw $refLevel0}
123.    ASA1 marker -action off
124.    if {$powerAsaRespRaw < -110} {
125.        Log h "$loggingTitle: AWG1 settings: state=[AWG1 getval state] arb_state=
[AWG1 getval arb_state] power=[AWG1 getval level] center=[AWG1 getval center]"
126.        Log h "$loggingTitle: ASA1 settings: center=[ASA1 getval center] span=[AS
A1 getval span] att=[ASA1 getval att] ref_level=[ASA1 getval ref_level]"
127.        Log h "$loggingTitle: Switch settings: SC01=[SC01 getval position]"
128.        error "MEAS_MCARRIER: no CW signal at $options(respPath) and $calGainResp
Freq MHz detected!"
129.    }
130.    if {abs($options(stimCenter) - $options(respCenter)) < 1} {set respOffset 0}
131.
132.    ## calibrate ASA against PWS1
133.    if {[string toupper $options(calAsaPwmResp)] == "AUTO"} {
134.        set calAsaPwmResp [calAsaPwm -center $calGainRespFreq -
respPath $options(respPath)]
135.    } else {
136.        set calAsaPwmResp $options(calAsaPwmResp)
137.    }
138.    set calGainTilt [expr {$powerAsaRespRaw - [$dlPathSa calcCalGain -
center $calGainRespFreq] - $calAsaPwmResp - $options(stimPower)}]
139.    Log t "Raw power: [format %.3f $powerAsaRespRaw] dBm at $calGainRespFreq MHz
,response offset: [format %.7f $respOffset] MHz"
140.    Log d "calAsaPwmResp=[format %.3f $calAsaPwmResp] calGainTilt=[format %.3f $
calGainTilt]"
141.    fb step $fbid

```

```

142.
143.     ##### measure again at calGainRespFreq2 for tilt corr
    action #####
144.     set calGainFreq2 [format %.1f [expr {$options(stimCenter) + $options(span) /
    4.}]]
145.     set calGainRespFreq2 [expr {$calGainFreq2 + $options(respCenter) - $options(
    stimCenter)}]
146.
147.     set pathFactor [$ulPath calcPathFactor -center $calGainFreq2]
148.     AWG1 setval center $calGainFreq2 state "ON"
149.     set difference [setMCPower -path $ulPath -center $calGainFreq2 -
    ulPower $options(stimPower) -pathFactor $pathFactor]
150.     ## switch signal to uplink if target power is achieved
151.     if {abs($difference) <= 0.1} {
152.         $ulPath setval state on
153.     } else {
154.         if {[valbool $::SCOE_CONFIG(startup.Simulated)]} {
155.             error "Target power $options(stimPower) +/-
    0.5 dBm could not be set! difference=$difference dB"
156.         } else {
157.             Log w "$loggingTitle: Simulated target power $options(stimPower) +/-
    0.1 dBm could not be set! difference=$difference dB"
158.         }
159.     }
160.     $dlPathSa setval state "ON"
161.     ## measure respPower with ASA
162.     ASA1 setval center $calGainRespFreq2 ;# same ASA settings
163.
164.     ASA1 marker -action {peak} -sweep yes
165.     set measurement [ASA1 getdata_marker -sweep yes]
166.     set powerAsaCalRaw2 [getOption $measurement -power]
167.     if {[ASA1 isSimulated]} {set powerAsaCalRaw2 $refLevel0}
168.     ASA1 marker -action off
169.
170.     ## calibrate ASA against PWS1
171.     if {[string toupper $options(calAsaPwmResp)] == "AUTO"} {
172.         set calAsaPwmTilt2 [calAsaPwm -center $calGainRespFreq2 -
    respPath $options(respPath)]
173.     } else {
174.         set calAsaPwmTilt2 $options(calAsaPwmResp)
175.     }
176.     set calGainTilt2 [expr {$powerAsaCalRaw2 - [$dlPathSa calcCalGain -
    center $calGainRespFreq2] - $calAsaPwmTilt2 - $options(stimPower)}]
177.     Log t "Raw power: [format %.3f $powerAsaCalRaw2] dBm at $calGainRespFreq2 MH
    z"
178.     Log d "calAsaPwmTilt2=[format %.3f $calAsaPwmTilt2] calGainTilt2=[format %.3
    f $calGainTilt2]"
179.
180.     if {[valbool $options(useCalSet)]} {
181.         MCAMPCALS clear
182.         catch {MCAMPCALS load}
183.         MCPHASECALs clear
184.         catch {MCPHASECALs load}
185.     }
186.     foreach stimCenter $stimCenterList span $spanList {
187.
188.         ## expand span by respSpanExpansion to even multiple of spacing

```

```

189.         set span2 [expr {2. * $spacing * round($span * (1. + [CONF getval respSp
anExpansion] / 100.) / ($spacing * 2.))}]
190.         ## expand span by stimSpanExpansion to even multiple of spacing, stimulu
s span is even wider than measurement span
191.         set spanStim [expr {2. * $spacing * round($span2 * (1. + [CONF getval st
imSpanExpansion] / 100.) / ($spacing * 2.))}]
192.         set start2 [format %.4f [expr {$spacing * round(($stimCenter - $spanStim
/ 2.) / $spacing)}]]
193.         set stop2 [format %.4f [expr {$spacing * round(($stimCenter + $spanStim
/ 2.) / $spacing)}]]
194.
195.         ## remember old respStop
196.         set respStopOld $respStop
197.         set respCenter [expr {$stimCenter - ($options(stimCenter) - $options(res
pCenter))}]
198.         set respStart [format %.4f [expr {$spacing * round(($respCenter - $span
/ 2.) / $spacing)}]]
199.         set respStop [format %.4f [expr {$spacing * round(($respCenter + $span
/ 2.) / $spacing)}]]
200.
201.         ## setup stimulus
202.         set sdesc "$loggingTitle: setup stimulus at $stimCenter MHz ..."
203.         set ::mainHelpvar $sdesc
204.         Log n $sdesc
205.
206.         set testPhase [CONF getval testPhase] ;## check if a debug file name is
specified in "Test phase"
207.         if {[string range $testPhase 0 2] != "PTS" || [string range $testPhase e
nd-3 end] != ".csv"} {
208.             ## start measurement, apply Chirp stimulus
209.             set parameters [list stimPath=$options(stimPath) carrierSpacing=$sp
acing startFreq=$start2 stopFreq=$stop2 stimPower=$options(stimPower)]
210.             set result [egseRunCommand -command_name SETUP_CHIRP -
parameter_string $parameters -handle_state no -handle_feedback no]
211.             set execCodes [scoe_server cget -execCodes]
212.             if {[getOption $result -
check_code] != [dict get $execCodes "noError"]} {error "Setting stimulus with SETUP_CHI
RP failed, wrong parameters=$parameters"}
213.             if {[getOption $result -
exec_code] != [dict get $execCodes "commandSuccess"]} {error "Setting stimulus with SET
UP_CHIRP failed, result= $result"}
214.             fb step $fbid
215.
216.             set sdesc "$loggingTitle: measuring response at $respCenter MHz..."
217.             set ::mainHelpvar $sdesc
218.             Log n $sdesc
219.             $dlPathSa setval state "ON"
220.             ## get optimum ASA settings
221.             ASA1 setval inst_mode "SAN" center $respCenter att $attGain ref_level
1 $refLevel0
222.             idle 200
223.             set asaSettings [calcAsaRefAtt -att $options(atten) -
path $dlPathSa -center $respCenter -preamp $options(preAmp)\
224.                 -reflevel $options(refLevel) -steps "CONF" -margin 2]
225.             set preAmp [getOption $asaSettings preamp_mode]
226.             set refLevel [max $refLevel0 [getOption $asaSettings ref_level]]
227.             set att [max 5 $att0 [getOption $asaSettings att]]

```



```

228.
229.         ## measure response:query phase and magnitude with IQ Analyser
230.         set result2 [ASA1.getdata_mcsignals_iq -
center [expr {$respCenter + $respOffset}] -span $span2 -spacing $spacing -att $att\
231.         -refLevel $refLevel -preAmp $preAmp -
avgcnt $options(avgCnt) -offset "EST" -trigger "EXT1" -traceAver $traceAver]
232.
233.         set respFreqs1 [getOption $result2 -frequency]
234.         set respAmpsRaw [getOption $result2 -power]
235.         set respPhases [getOption $result2 -phase]
236.         check_events; fb step $fbid
237.
238.         if {[valbool $options(useCalSet)]} {
239.             ## try to get calibration data set, these are calibrated values
at the test bracket for 0dBm stim power
240.             set calId [list stimFreq [format %.3f $stimCenter] respFreq [for
mat %.3f $respCenter] span [format %.3f $span] refLevel [format %.3f $refLevel]]
241.             set calItem [MCAMPCALS getChild $calId]
242.             if {$calItem == {}} {error "No suitable AMP calset found! calId=
$calId"} else {Log t "calset found: calId=$calId"}
243.             set calFreqs [$calItem getval indata]
244.             set calAmps [$calItem getval outdata]
245.
246.             ## check length of frequency list, because spanExpansion could h
ave been changed
247.             set freqCount [expr {int(round(1 + $span2 / ($spacing)))]}
248.             if {$freqCount != [llength $calFreqs]} {error "No suitable calse
t found!\
249.             calId=$calId, llength calFreqs = [llength $calFreqs] req
uired frequency count = $freqCount"}
250.             ## get calibration phases
251.             set calItem [MCPHASECALs getChild $calId]
252.             if {$calItem == {}} {error "No suitable PHASE calset found! calI
d=$calId"}
253.             set calPhases [$calItem getval outdata]
254.         } else {
255.             ## measure stimulus if no calset has to be used
256.             set sdesc "$loggingTitle: measuring stimulus at $stimCenter MHz.
.."
257.             set ::mainHelpvar $sdesc
258.             Log n $sdesc
259.
260.             ## switch path for stimulus measurement
261.             $ulPath setval state "OFF"
262.             SC01 setval position 6
263.             SC02 setval position 2
264.
265.             ## check if power at ASA is sufficient
266.             set asaPower [expr {$options(stimPower) - [$ulPath calcPathFacto
r -center $options(stimCenter)] \
267.             - [$dlPathPm calcCalGain -
center $options(stimCenter)] + [$dlPathSa calcCalGain -center $options(stimCenter)]}]
268.
269.             if {($asaPower < -30)} {
270.                 set deltaPower [expr {10 - 10 * int((40. + $asaPower) / 10.)
}]
271.                 Log t "AWG power too low for stimulus measurement! AWG power
increased by $deltaPower"

```

```

272.                AWG1 setval level [expr {[AWG1 getval level] + $deltaPower}]
273.                } else {set deltaPower 0}
274.
275.                ASA1 preset
276.                ASA1 setval center $stimCenter
277.                ASA1 write "TRIG:SOUR IMM"
278.                idle 200
279.                set asaSettings [calcAsaRefAtt -att $options(atten) -
280.                path $options(stimPath) -center $stimCenter -preamp $options(preAmp)\
                -reflevel $options(refLevel) -steps 1 -margin 5 -
                pwr $asaPower]
281.
282.                set preAmp [getOption $asaSettings preamp_mode]
283.                set refLevels [getOption $asaSettings ref_level]
284.                set att [getOption $asaSettings att]
285.
286.                ## stimulus channel power is taken from options(stimPower), not
                measured
287.                ## query phase and magnitude with IQ Analyser
288.                set result1 [ASA1.getdata_mcsignals_iq -center $stimCenter -
                span $span2 -spacing $spacing\
289.                -att $att -refLevel $refLevels -
                preAmp $preAmp -avgcnt $options(avgCnt) -offset "EST" -trigger "EXT1" -
                traceAver $traceAver]
290.
291.                set calFreqs [getOption $result1 -frequency]
292.                set calAmps [getOption $result1 -power]
293.                set calPhases [getOption $result1 -phase]
294.                check_events
295.
296.                ## calibrate amps
297.                set calAmps [getCalibratedPowerFromRawPower -
                freqList $calFreqs -powerList $calAmps -path $ulPath -instrument "ASA"
298.                fb step $fbid
299.                }
300.                if {[CONF getval testPhase] == "debug"} {
301.                    set respFreqs$stimCenter $respFreqs1
302.                    set respAmps$stimCenter $respAmpsRaw
303.                    set respPhases$stimCenter $respPhases
304.                    set calAmps$stimCenter $calAmps
305.                    set calPhases$stimCenter $calPhases
306.                }
307.                } else {
308.                    ## no measurement, but read raw data from debug file (eg PTSTCR11-
                    20181123-201912.csv), name is configured as "Test phase" in CONF
309.                    ## call with calAsaPwmResp=0 !! set in Conf namespace maxSpan compat
                    ible with the debug file
310.                    set respFreqs1 {}; set respAmpsRaw {}; set respPhases {}; set calAmps
                    s {}; set calPhases {}
311.                    set pathName [file join $::SPOOL_PATH "Results" $testPhase]
312.                    Log h "No measurement, reading from $pathName!"
313.                    set fd [open $pathName r]
314.                    while {[eof $fd]} {set line [gets $fd]; if {[lindex $line end] == [
                    format %.1f $stimCenter]} {set line1 $line; break}}
315.                    gets $fd ;## skip header line
316.                    while {[eof $fd]} {
317.                        set line [gets $fd]; if {[regexp "#" $line]} {break}

```

```

318.             lappend respFreqs1 [lindex $line 0]; lappend respAmpsRaw [lindex
319. $line 1]
319.             lappend respPhases [lindex $line 3]; lappend calAmps [lindex $li
320. ne 2]
320.             lappend calPhases [lindex $line 4]
321.         }
322.         close $fd
323.         Log t "stimCenter found in file: $line1, [length $respFreqs1] frequ
324. encies read in, first=[lindex $respFreqs1 0] last=[lindex $respFreqs1 end]"
324.         set autotest 0; set deltaPower 0.; set gdCorr 0.
325.         set respFreqs$stimCenter $respFreqs1
326.         set respAmps$stimCenter $respAmpsRaw
327.         set respPhases$stimCenter $respPhases
328.         set calAmps$stimCenter $calAmps
329.         set calPhases$stimCenter $calPhases
330.         fb step $fbid 2
331.     }
332.
333.     ## calibrate measured amplitudes
334.     set respAmpsCal [getCalibratedPowerFromRawPower -freqList $respFreqs1 -
335. powerList $respAmpsRaw -path $dlPathSa -instrument "ASA"]
335.     ## calculate amplitude differences:
336.     set ampDiffs1 {}
337.
338.     if {[valbool $options(useCalSet)]} {
339.         ## calAmps are normalized to 0dBm total for TAC, to 10*log(length)
340.         for autotestrack = 0dBm per line
341.         ## check autotest rack calibration
341.         set middle [expr {int([length $calAmps] / 2)}]
342.         if {[lindex $calAmps $middle] > -12} {
343.             set stimPowerCorr [expr {$options(stimPower) - 10. * log10([llen
344. gth $calAmps]}]}
344.             set autotest 1
345.         } else {
346.             set stimPowerCorr $options(stimPower)
347.             set autotest 0
348.         }
349.         foreach st $calAmps rs $respAmpsCal {lappend ampDiffs1 [expr {$rs -
350. $st - $stimPowerCorr]}]}
350.     } else {
351.         if {$ampDiffs == {}} {
352.             ## correct gain for first span
353.             ## sum up total power measured with K7 and compare it with optio
354. ns(stimPower)
354.             set k7StimPower 0
355.             foreach p $calAmps {set k7StimPower [expr {$k7StimPower + pow(10
356. ., $p / 10.)}}]}
356.             set k7StimPower [expr {10. * log10($k7StimPower)}]
357.             set deltaStim [expr {$options(stimPower) - $k7StimPower}]
358.             set stimAmps {}
359.             foreach a $calAmps {lappend stimAmps [expr {$a + $deltaStim}]}
360.             set calAmps $stimAmps
361.             Log d "$loggingTitle: k7StimPower=$k7StimPower correction delta
362. Stim=[format %.3f $deltaStim]"
362.         }
363.         foreach st $calAmps rs $respAmpsCal {lappend ampDiffs1 [expr {$rs -
364. $st - $deltaPower}]}

```

```

364.         }
365.
366.         ## calculate phase differences:
367.         set phaseDiffs1 {}
368.         foreach sp $calPhases rp $respPhases {lappend phaseDiffs1 [expr {$rp - $
    sp}}]
369.         set phaseDiffs1 [phase_continuous $phaseDiffs1]
370.
371.         ## eliminate values at vsgCenter: read vsgCenter from VSG, search index
    of center frequency
372.         set vsgCenter [AWG1 getval center]
373.
374.         if {abs($vsgCenter - $stimCenter) <= $span2 / 2.} {
375.             set vsgCenter [expr {$vsgCenter - ($options(stimCenter) -
    $options(respCenter))}]
376.             set idxCenter [expr {round(($vsgCenter - [lindex $respFreqs1 0]) / $
    spacing)}]
377.             if {$idxCenter == 0} {
378.                 lset ampDiffs1 $idxCenter [expr {(2. * [lindex $ampDiffs1 1] -
    [lindex $ampDiffs1 2])}]
379.                 lset phaseDiffs1 $idxCenter [expr {(2. * [lindex $phaseDiffs1 1]
    - [lindex $phaseDiffs1 2])}]
380.             } elseif {$idxCenter == ([llength $respFreqs1] - 1)} {
381.                 lset ampDiffs1 $idxCenter [expr {(2. * [lindex $ampDiffs1 end-
    1] + [lindex $ampDiffs1 end-2])}]
382.                 lset phaseDiffs1 $idxCenter [expr {(2. * [lindex $phaseDiffs1 en
    d-1] + [lindex $phaseDiffs1 end-2])}]
383.             } elseif {($idxCenter > 0) && ($idxCenter < [llength $respFreqs1])}
    {
384.                 lset ampDiffs1 $idxCenter [expr {[lindex $ampDiffs1 $idxCente
    r-1] + [lindex $ampDiffs1 $idxCenter+1]} / 2.}]
385.                 lset phaseDiffs1 $idxCenter [expr {[lindex $phaseDiffs1 $idxCen
    ter-1] + [lindex $phaseDiffs1 $idxCenter+1]} / 2.}]
386.             }
387.             Log d "$loggingTitle: Value of VSG center at $vsgCenter MHz substitu
    ted by interpolation"
388.         }
389.         ## append to previous frequency list, respFreqs is empty for first measu
    rement
390.
391.         if {$respFreqs == {}} {
392.             set respFreqs $respFreqs1
393.         } else {
394.
395.             ## cut spanExpansion frequency range from end of previous respFreqs
    list
396.             for {set i [expr {[llength $respFreqs] - 1]} {$i > 0} {incr i -
    1} {
397.                 if {[lindex $respFreqs $i] - $respStopOld < $spacing / 100.} {br
    eak}
398.             }
399.             set respFreqs [lrange $respFreqs 0 $i]
400.             ## cut spanExpansion frequency range from start of current respFreqs
    list
401.             for {set j 0} {$j < [expr {[llength $respFreqs1] - 1]} {incr j} {
402.                 if {[lindex $respFreqs1 $j] - [lindex $respFreqs end] > $spacing
    / 100.} {break}

```

```

403.         }
404.         set respFreqs1 [lrange $respFreqs1 $j end]
405.         set respFreqs [concat $respFreqs $respFreqs1]
406.         set avgSpan [expr {int(2*$j)}] ;# number of samples in overlapping r
ange used for aligning
407.         }
408.         ## append to previous amp list with offset
409.         if {$ampDiffs == {}} {
410.             set ampDiffs $ampDiffs1
411.         } else {
412.             ## calculate average values over +- expansion span
413.             Log d "MEAS_MCARRIER: interpolation interval i=$i j=$j avgSpan=$avg
Span"
414.             set avgDiffs 0
415.             foreach ad [lrange $ampDiffs [expr {[length $ampDiffs] - $avgSpan +
1}] end] {set avgDiffs [expr {$avgDiffs + $ad}]}
416.             set avgDiffs [expr {$avgDiffs / ($avgSpan - 1.)}]
417.             set avgDiffs1 0
418.             foreach ad [lrange $ampDiffs1 0 $avgSpan-
2] {set avgDiffs1 [expr {$avgDiffs1 + $ad}]}
419.             set avgDiffs1 [expr {$avgDiffs1 / ($avgSpan - 1.)}]
420.             set ampOffset [expr {$avgDiffs - $avgDiffs1}]
421.             Log d "avgDiffs=$avgDiffs avgDiffs1=$avgDiffs1 ampOffset = $ampOffs
et dB"
422.             set ampDiffs [lrange $ampDiffs 0 $i]
423.             set ampDiffs1 [lrange $ampDiffs1 $j end]
424.             foreach ad $ampDiffs1 {lappend ampDiffs [expr {$ad + $ampOffset}]}
425.         }
426.
427.         ## append to previous phase list with offset and slope offset
428.         if {$phaseDiffs == {}} {
429.             set phaseDiffs $phaseDiffs1 ;# first span
430.         } else {
431.             ## filterCount = max. number of deviating phase values which will be
discarded for slope calculation
432.             for {set filterCount [min 4 [expr {$avgSpan / 2}]]} {$filterCount <
[expr {int($avgSpan / 2)}]} {incr filterCount 2} {
433.                 set res [linear_regression_filtered -count $filterCount -
yvalues [lrange $phaseDiffs [expr {[length $phaseDiffs] - $avgSpan + 1}] end]]
434.                 set slope [getOption $res -slope]
435.                 set extrapolationValue [expr {[getOption $res -
intercept] + $slope * $j}]
436.                 set res [linear_regression_filtered -count $filterCount -
yvalues [lrange $phaseDiffs1 0 $avgSpan-2]]
437.                 set slope1 [getOption $res -slope]
438.                 set extrapolationValue1 [expr {[getOption $res -
intercept] + $slope1 * $j}]
439.                 set phaseOffset [expr {$extrapolationValue - $extrapolationValue
1}]
440.                 set slopeDiff [expr {$slope1 - $slope}]
441.                 if {$span2 > 60} {set maxSlopeDiff 5} else {set maxSlopeDiff 0.0
5} ;# higher trigger uncertainty with spans >60MHz
442.                 if {$slopeDiff < $maxSlopeDiff} {break} ;## stop if difference i
s < maxSlopeDiff
443.             }
444.             Log d "slopeDiff = [format %.3f $slopeDiff] corresponding with [for
mat %.3f [expr {$slopeDiff * 25. / 9}]]ns discarded samples=$filterCount"
445.             set phaseDiffs [lrange $phaseDiffs 0 $i]

```

```

446.         set phaseDiffs1 [lrange $phaseDiffs1 $j end]
447.         set x 0
448.         foreach pd $phaseDiffs1 {
449.             lappend phaseDiffs [expr {$pd + $phaseOffset - $x * $slopeDiff }
]; incr x
450.         }
451.     }
452. } ;##### end of loop: foreach stimCenter $stimCenterList...
453.
454.
455.     if {[CONF getval gdAbsRel] == "ABS" && [valbool $options(useCalSet)]} {
456.
457.         ## calculate gdCorr = group delay offset of TAC or autotest rack, read f
rom CONF and correct it for center frequency
458.         if {$autotest} {set phaseCalCable "CABLE_AUTOTEST_PHASE"} else {set phas
eCalCable "CABLE_TAC_PHASE"}
459.         if {$options(stimCenter) > 20000} {
460.             if {$autotest} {set gdCorr [CONF getval gdoCalFwdAuto]} else {set gd
Corr [CONF getval gdoCalFwd]}
461.             set fcalstart 0; set fcalstop 12800; set fcalcenter 11800
462.         } else {
463.             if {$autotest} {set gdCorr [CONF getval gdoCalRtnAuto]} else {set gd
Corr [CONF getval gdoCalRtn]}
464.             set fcalstart 17700; set fcalstop 20000; set fcalcenter 18530
465.         }
466.         ## build phase calibration table of TAC or AUTOTEST calibration
467.         if {[catch {set calObj [$phaseCalCable cget -calPhaseObj]}]} {
468.             error "no phase calibration for $phaseCalCable found!"
469.         } else {
470.             ## make the calibration phase values continuous
471.             set calFreqList {}; set calPhaseList {}
472.             foreach f [$calObj getval indata] p [$calObj getval outdata] {
473.                 if {$f >= $fcalstart && $f <= $fcalstop} {
474.                     lappend calFreqList $f
475.                     lappend calPhaseList $p
476.                 }
477.             }
478.             set calPhaseList [phase_continuous $calPhaseList]
479.             ## calculate groupDelayList of TAC: group delay = phase slope / delt
a frequency, slope is calculated over 1MHz intervall
480.             set gdspan [max 2 [expr {int(round(4 / ([lindex $calFreqList 1] - [l
index $calFreqList 0]))}]]]
481.             set calGroupDelayList [differentiateList -xvalues $calFreqList -
yvalues $calPhaseList -span $gdspan]
482.             ## conversion factor from degrees to radians and from MHz to Hz and
to ns
483.             set factor [expr {-25. / 9.}]
484.             for {set i 0} {$i < [llength $calGroupDelayList]} {incr i} {
485.                 lset calGroupDelayList $i [expr {$factor * [lindex $calGroupDela
yList $i]}]
486.             }
487.             ## search GD of TAC in the middle and GD at resp center and calculat
e difference
488.             set calGdTable {}
489.             foreach freq $calFreqList ph $calGroupDelayList {lappend calGdTable
    $freq $ph}
490.             set gdRespCenter [format %.3f [::math::interpolate::interp-
linear $calGdTable $options(respCenter)]]

```

```

491.         set gdCalCenter [format %.3f [::math::interpolate::interp-
linear $calGdTable $fcalcenter]]
492.         set gdCorr [format %.3f [expr {$gdCorr + $gdRespCenter - $gdCalCenter}]]
493.         Log t "GD of TAC @ [format %.0f $options(respCenter)]= $gdRespCenter
ns GD of TAC @ $fcalcenter = $gdCalCenter ns correction=$gdCorr ns"
494.         }
495.
496.         ## measure again with 50MHz span for absolute group delay
497.         if {$span > 60 && [valbool $options(useCalSet)]} {
498.             set spanC 50
499.             ## expand spanC by respSpanExpansion to even multiple of spacing
500.             set span2 [expr {2. * $spacing * round($spanC * (1. + [CONF getval respSpanExpansion] / 100.) / ($spacing * 2.))}]
501.             ## expand spanC by stimSpanExpansion to even multiple of spacing, stimulus span is even wider than measurement span
502.             set spanStim [expr {2. * $spacing * round($span2 * (1. + [CONF getval stimSpanExpansion] / 100.) / ($spacing * 2.))}]
503.             set start2 [format %.4f [expr {$spacing * round(($stimCenter - $spanStim / 2.) / $spacing)}]]
504.             set stop2 [format %.4f [expr {$spacing * round(($stimCenter + $spanStim / 2.) / $spacing)}]]
505.
506.             set respCenter [expr {$stimCenter - ($options(stimCenter) - $options(respCenter))}]
507.             set respStart [format %.4f [expr {$spacing * round(($respCenter - $spanC / 2.) / $spacing)}]]
508.             set respStop [format %.4f [expr {$spacing * round(($respCenter + $spanC / 2.) / $spacing)}]]
509.
510.             ## setup stimulus
511.             set sdesc "$loggingTitle: setup stimulus at $stimCenter MHz with 50M
Hz span..."
512.             set ::mainHelpvar $sdesc
513.             Log n $sdesc
514.
515.             ## apply Chirp stimulus
516.             set parameters [list stimPath=$options(stimPath) carrierSpacing=$spacing startFreq=$start2 stopFreq=$stop2 stimPower=$options(stimPower)]
517.             set result [egseRunCommand -command_name SETUP_CHIRP -
parameter_string $parameters -handle_state no -handle_feedback no]
518.             set execCodes [scoe_server cget -execCodes]
519.             if {[getOption $result -
check_code] != [dict get $execCodes "noError"]} {error "Setting stimulus with SETUP_CHIRP failed, wrong parameters=$parameters"}
520.             if {[getOption $result -
exec_code] != [dict get $execCodes "commandSuccess"]} {error "Setting stimulus with SETUP_CHIRP failed, result= $result"}
521.             fb step $fbid
522.
523.             set sdesc "$loggingTitle: measuring response at $respCenter MHz..."
524.             set ::mainHelpvar $sdesc
525.             Log n $sdesc
526.             $dlPathSa setval state "ON"
527.
528.             ## measure response: measure MC signa with IQ Analyser

```



```

529.         set result2 [ASA1.getdata_mcsignals_iq -
center [expr {$respCenter + $respOffset}] -span $span2 -spacing $spacing -att $att\
530.         -refLevel $refLevel -preAmp $preAmp -
avgcnt $options(avgCnt) -offset "EST" -trigger "EXT1" -traceAver $traceAver]
531.
532.         set respFreqsC [getOption $result2 -frequency]
533.         set respPhasesC [getOption $result2 -phase]
534.         check_events
535.         fb step $fbid
536.
537.         ## try to get calibration data set, these are calibrated values at t
he test bracket for 0dBm stim power
538.         set calId [list stimFreq [format %.3f $stimCenter] respFreq [format
%.3f $respCenter] span [format %.3f $spanC] refLevel [format %.3f $refLevel]]
539.         set calItem [MCAMPCALS getChild $calId]
540.         if {$calItem == {}} {error "No suitable AMP calset found! calId=$cal
Id"} else {Log t "calset found: calId=$calId"}
541.         set calFreqs [$calItem getval indata]
542.         set calAmps [$calItem getval outdata]
543.
544.         ## check length of frequency list, because spanExpansion could have
been changed
545.         set freqCount [expr {int(round(1 + $span2 / ($spacing)))]}
546.         if {$freqCount != [llength $calFreqs]} {error "No suitable calset fo
und!\
547.         calId=$calId, llength calFreqs = [llength $calFreqs] require
d frequency count = $freqCount"}
548.         ## get calibration phases
549.         set calItem [MCPHASECALS getChild $calId]
550.         if {$calItem == {}} {error "No suitable PHASE calset found! calId=$c
alId"}
551.         set calPhases [$calItem getval outdata]
552.
553.         ## calculate phase differences:
554.         set phaseDiffsC {}
555.         foreach sp $calPhases rp $respPhasesC {lappend phaseDiffsC [expr {$r
p - $sp}]}
556.         set phaseDiffsC [phase_continuous $phaseDiffsC]
557.
558.         ## eliminate values at vsgCenter: read vsgCenter from VSG, search in
dex of center frequency
559.         set vsgCenter [AWG1 getval center]
560.
561.         if {abs($vsgCenter - $stimCenter) <= $span2 / 2.} {
562.             set vsgCenter [expr {$vsgCenter - ($options(stimCenter) -
$options(respCenter))}]
563.             set idxCenter [expr {round(($vsgCenter - [lindex $respFreqsC 0])
/ $spacing)}]
564.             if {$idxCenter == 0} {
565.                 lset phaseDiffsC $idxCenter [expr {(2. * [lindex $phaseDiffs
C 1] - [lindex $phaseDiffsC 2])}]
566.             } elseif {$idxCenter == ([llength $respFreqs1] - 1)} {
567.                 lset phaseDiffsC $idxCenter [expr {(2. * [lindex $phaseDiffs
C end-1] + [lindex $phaseDiffsC end-2])}]
568.             } elseif {($idxCenter > 0) && ($idxCenter < [llength $respFreqs1
])} {
569.                 lset phaseDiffsC $idxCenter [expr {( [lindex $phaseDiffsC $id
xCenter-1] + [lindex $phaseDiffsC $idxCenter+1]) / 2.}]

```

```

570.         }
571.         Log d "$loggingTitle: Value of VSG center at $vsgCenter MHz subs
tituted by interpolation"
572.     }
573.
574.     ## calculate groupDelayList: group delay = phase slope / delta frequ
ency, slope is calculated over 1MHz intervall
575.     set groupDelayListC [differentiateList -xvalues $respFreqsC -
yvalues $phaseDiffsC -span $spanC]
576.     ## conversion factor from degrees to radians and from MHz to Hz and
to ns
577.     set factor [expr {-25. / 9.}]
578.     for {set i 0} {$i < [llength $groupDelayListC]} {incr i} {
579.         lset groupDelayListC $i [expr {$factor * [lindex $groupDelayList
C $i]}]
580.     }
581.
582.     ## calculate average group delay in the middle from iGDStart to IGDS
top
583.     set iGDStart [expr {int(0.33 * [llength $groupDelayListC])}]
584.     set iGDStop [expr {int(0.66 * [llength $groupDelayListC])}]
585.     set freqGDStart [lindex $respFreqsC $iGDStart]
586.     set freqGDStop [lindex $respFreqsC $iGDStop]
587.     set gdAverC 0
588.     for {set i $iGDStart} {$i < $iGDStop} {incr i} {
589.         set gdAverC [expr {$gdAverC + [lindex $groupDelayListC $i]}]
590.     }
591.     set gdOffsetRawC [expr {$gdAverC / ($iGDStop - $iGDStart)}]
592.     set gd50MHzSpan [expr {$gdOffsetRawC + $gdCorr}]
593.     Log d "gd cal measurement uncorrected = [format %.3f $gdOffsetRawC]
gdCorr=$gdCorr gd50MHzSpan=[format %.3f $gd50MHzSpan]"
594.     }
595.     } else {
596.         ## relative group delay
597.         set gd50MHzSpan 0.
598.     }
599.
600.     ASA1 setval sweep_mode CONT
601.
602.     ##### gain and tilt correction #####
#####
603.     ## first smooth amps
604.     set smoothing [expr {100. * ($options(aperture) / $spacing) / [llength $ampD
iffs]}]
605.     set ampDiffs1 [smoothTrace -trace $ampDiffs -smoothing $smoothing]
606.
607.     ## apply gain correction, search for indices of calGainFreq1 and calGainFreq
2
608.     if {[string toupper $options(calAsaPwmResp)] == "AUTO"} {
609.         ## find index for calGainRespFreq
610.         for {set idxTilt [expr {int([llength $respFreqs] * 0.25)}]} {$idxTilt >
0 && $idxTilt < [llength $respFreqs]} {} {
611.             if {abs([lindex $respFreqs $idxTilt] - $calGainRespFreq) < $spacing
/ 2.} {break}
612.             if {[lindex $respFreqs $idxTilt] < $calGainRespFreq} {incr idxTilt}
else {incr idxTilt -1}
613.         }
614.         set deltaGain [expr {[lindex $ampDiffs1 $idxTilt] - $calGainTilt}]

```

```

615.         Log d "$loggingTitle: idxTilt=$idxTilt freq=[lindex $respFreqs $idxTilt
] deltaGain=$deltaGain"
616.
617.         ## find index for calGainRespFreq2
618.         for {set idxTilt2 [expr {int([llength $respFreqs] * 0.75)}]} {$idxTilt2
> 0 && $idxTilt2 < [llength $respFreqs]} {} {
619.             if {abs([lindex $respFreqs $idxTilt2] - $calGainRespFreq2) < $spacin
g / 2.} {break}
620.             if {[lindex $respFreqs $idxTilt2] < $calGainRespFreq2} {incr idxTilt
2} else {incr idxTilt2 -1}
621.         }
622.         set deltaGain2 [expr {[lindex $ampDiffs1 $idxTilt2] - $calGainTilt2}]
623.         Log d "$loggingTitle: idxTilt2=$idxTilt2 freq=[lindex $respFreqs $idxTi
lt2] deltaGain2=$deltaGain2"
624.         set kTilt [expr {($deltaGain2 - $deltaGain) / ($idxTilt2 - $idxTilt)}]
625.         set dTilt [expr {$deltaGain - $idxTilt * $kTilt}]
626.
627.         } else {
628.             Log d "$loggingTitle: options(calAsaPwmResp)=$options(calAsaPwmResp) ...
no tilt correction ... gain correction by $calAsaPwmResp dB"
629.             set kTilt 0
630.             set dTilt $calAsaPwmResp
631.         }
632.
633.         ## apply tilt and gain correction
634.         set ampDiffs {}
635.         for {set i 0} {$i < [llength $ampDiffs1]} {incr i} {
636.             lappend ampDiffs [expr {[lindex $ampDiffs1 $i] - $kTilt * $i - $dTilt}]
637.         }
638.         set indx [expr {[llength $ampDiffs] / 2}]
639.         Log d "$loggingTitle: ampDiffs = [lindex $ampDiffs $indx] ... f = [lindex $r
espFreqs $indx]"
640.
641.         ## calculate gainSlopeList: linear regression over 1MHz sliding window, span
is number of carriers for slope calculation
642.         set gainSlopeList {}
643.         if {$options(aperture) > $options(span)} {set options(aperture) $options(spa
n)}
644.         set spanA [max 2 [expr {round($options(aperture) / ($spacing))}] ]
645.         set gainSlopeList [differentiateList -xvalues $respFreqs -
yvalues $ampDiffs -span $spanA]
646.
647.         ## remove outliers from phaseDiffs
648.         set smoothing [max 1 [min 50 [expr {100. / $options(span)}]]]
649.         if {[ASA1 isSimulated]} {set smoothing 0.1}
650.         set res [removeOutliers -trace $phaseDiffs -threshold 5 -
smoothing $smoothing]
651.         #set phaseDiffs [getOption $res -traceWoOutliers]
652.         Log t "$loggingTitle: Phase smoothing = [format %.2f $smoothing] ... Outlier
Anzahl = [llength [getOption $res -outlierIndexList]]"
653.
654.         ## calculate groupDelayList: group delay = phase slope / delta frequency, sl
ope is calculated over aperture MHz intervall
655.         set groupDelayList [differentiateList -xvalues $respFreqs -
yvalues [getOption $res -traceWoOutliers] -span $spanA]
656.         ## check for maximum gd and try phase_continuous again, safety check
657.         set maxGdDiff [expr {[lmax $groupDelayList] - [lmin $groupDelayList]}]

```

```

658.         if {$maxGdDiff > 80} {
659.             set phaseDiffs [phase_continuous90 $phaseDiffs]
660.             set res [removeOutliers -trace $phaseDiffs -threshold 5 -
smoothing $smoothing]
661.             set phaseDiffs [getOption $res -traceWoOutliers]
662.             set groupDelayList [differentiateList -xvalues $respFreqs -
yvalues [getOption $res -traceWoOutliers] -span $spanA]
663.             Log t "$loggingTitle: maxGdDiff=[expr {25. * $maxGdDiff / 9.}] repeat ph
ase_continuous"
664.             set maxGdDiff [expr {[lmax $groupDelayList] - [lmin $groupDelayList]}]
665.             Log t "$loggingTitle: maxGdDiff=[expr {25. * $maxGdDiff / 9.}] after rep
eated phase_continuous"
666.         } else {
667.             set phaseDiffs [getOption $res -traceWoOutliers]
668.         }
669.
670.         ## conversion factor from degrees to radians and from MHz to rad/s and from
seconds to ns
671.         set factor [expr {-25. / 9.}]
672.         for {set i 0} {$i < [llength $groupDelayList]} {incr i} {
673.             lset groupDelayList $i [expr {$factor * [lindex $groupDelayList $i]}]
674.         }
675.         ## calculate group delay in the middle
676.         set gdListMiddle [lrange $groupDelayList [expr {int(0.45*[llength $groupDela
yList])}] [expr {int(0.55*[llength $groupDelayList])}]]
677.         set gdOffsetRaw [expr {[lsum $gdListMiddle] / [llength $gdListMiddle]}]
678.
679.         if {$span > 60 && [valbool $options(useCalSet)]} {
680.             set gdCorr [expr {$gd50MHzSpan - $gdOffsetRaw}]
681.         }
682.         if { [CONF getval gdAbsRel] != "ABS" || ![valbool $options(useCalSet)]} {
683.             ## only relative group delay, normalize to zero
684.             set gdCorr [expr {-1. * $gdOffsetRaw}]
685.         }
686.
687.         Log t "uncorrected group delay offset [format %.2f $gdOffsetRaw] ns, correct
ed by [format %.2f $gdCorr] ns"
688.
689.         ## calculate testcap group delays
690.         if {[valbool [CONF getval enableTestcaps]]} {
691.             set temperature [CONF getval testcapTemperature]
692.             if {[regexp "KUUL" $options(stimPath)]} {
693.                 set testCapCable TESTCAP_TEMP($temperature)_[lindex [split $options(
stimPath) "-"] 0]
694.                 set testCapObj [$testCapCable cget -calDelayObj]
695.             } elseif {[regexp "KUDL" $options(respPath)]} {
696.                 set testCapCable TESTCAP_TEMP($temperature)_[lindex [split $options(
respPath) "-"] 0]
697.                 set testCapObj [$testCapCable cget -calDelayObj]
698.             }
699.             set testCapDelayTable {}
700.             if {[regexp "KUUL" $options(stimPath)]} {set loFreq [expr {$options(resp
Center) - $options(stimCenter)}]} else {set loFreq 0}
701.             foreach freq [$testCapObj getval indata] delay [$testCapObj getval outda
ta] {
702.                 lappend testCapDelayTable [expr {$freq + $loFreq}] $delay
703.             }

```

```

704.          Log d "MEAS_MCARRIER: delay of $testCapCable calculated"
705.          for {set i 0} {$i < [llength $groupDelayList]} {incr i} {
706.              set gdTestCap [::math::interpolate::interp-
linear $testCapDelayTable [lindex $respFreqs $i]]
707.              lset groupDelayList $i [expr {[lindex $groupDelayList $i] + $gdCorr
- $gdTestCap}]
708.          }
709.      } else {
710.          for {set i 0} {$i < [llength $groupDelayList]} {incr i} {
711.              lset groupDelayList $i [expr {[lindex $groupDelayList $i] + $gdCorr}
]
712.          }
713.      }
714.
715.      ## remove values for frequency range < center-
span/2 and > center+span/2 from result lists
716.      for {set i 0} {$i < [expr {[llength $respFreqs] - 1}]} {incr i} {
717.          if {[lindex $respFreqs $i] > [expr {$options(respCenter) - ($options(spa
n) / 2.) + ($spacing / 2.)}]} {break}
718.      }
719.      incr i -1
720.      ## cut frequencies at the end
721.      for {set j [expr {[llength $respFreqs] - 1}]} {$j > 0} {incr j -1} {
722.          if {[lindex $respFreqs $j] < [expr {$options(respCenter) + ($options(spa
n) / 2.) - ($spacing / 2.)}]} {break}
723.      }
724.      incr j
725.
726.      set respFreqs [lrange $respFreqs $i $j]
727.      set ampDiffs [lrange $ampDiffs $i $j]
728.      set groupDelayList [lrange $groupDelayList $i $j]
729.      set gainSlopeList [lrange $gainSlopeList $i $j]
730.      set phaseDiffs [lrange $phaseDiffs $i $j]
731.
732.      if {$evaluate == 1} {
733.          ## trim gain slope and flatness to evaluation bandwidth if specified
734.          set evalStart [expr {$options(evalCenter) - $options(evalSpan) / 2.}]
735.          set evalStop  [expr {$options(evalCenter) + $options(evalSpan) / 2.}]
736.          ## find eval start and eval stop indices in respFreqs list
737.          for {set e1 0} {$e1 < [llength $respFreqs]} {incr e1} { if {[lindex $r
espFreqs $e1] > $evalStart} {break}}
738.          for {set e2 $e1} {$e2 < [llength $respFreqs]} {incr e2} { if {[lindex $r
espFreqs $e2] > $evalStop} {break}}
739.          incr e2 -1
740.          set gainSlopeListEval [lrange $gainSlopeList $e1 $e2]
741.          set ampDiffListEval  [lrange $ampDiffs $e1 $e2]
742.          set gainSlope [lmax $gainSlopeListEval]
743.          set gainFlatness [expr {[lmax $ampDiffListEval] - [lmin $ampDiffListEval
]]}
744.          set phaseLinList [phase_lin_error -phaseList $phaseDiffs -
indexStart $e1 -indexStop $e2]
745.      } else {
746.          set gainSlope [lmax $gainSlopeList]
747.          set gainFlatness [expr {[lmax $ampDiffs] - [lmin $ampDiffs]}]
748.          set phaseLinList [phase_lin_error -phaseList $phaseDiffs]
749.      }
750.      fb step $fbid
751.

```

```

752.         ## get database id
753.         set dbid [$dbIfObj openResult -type [string tolower $cmdName] -
tag $options(respPath) -customid1 [CONF getval customId1] -
customid2 [CONF getval customId2] -sdesc "$loggingTitle: ..."]
754.         ## store database id for outside access
755.         setval dbid [getOption $dbid -resultCommon.id]
756.         ## insert NObjects into db
757.         attachNObject -dbid $dbid -object CONF -async yes
758.         attachNObject -dbid $dbid -object $this -components [getcompOption -
!idlist result] -tag c&c -async yes
759.
760.         ## reduce list lengths to result count values and write results
761.
762.         set result [decimate2ResultCount -xvalues $respFreqs -
yvalues $gainSlopeList -count $options(resultCount)]
763.         set gainSlopeList [getOption $result -yvalues]
764.
765.         set result [decimate2ResultCount -xvalues $respFreqs -
yvalues $phaseLinList -count $options(resultCount)]
766.         set phaseLinList [getOption $result -yvalues]
767.
768.         set result [decimate2ResultCount -xvalues $respFreqs -
yvalues $groupDelayList -count $options(resultCount)]
769.         set groupDelayList [getOption $result -yvalues]
770.
771.         set result [decimate2ResultCount -xvalues $respFreqs -yvalues $ampDiffs -
count $options(resultCount)]
772.         set respFreqs [getOption $result -xvalues]
773.         set ampDiffs [getOption $result -yvalues]
774.
775.         set calStim [format %.3f [$ulPath calcCalGain -
center $options(stimCenter)]]
776.         set calResp [format %.3f [$dlPathSa calcCalGain -
center $options(respCenter)]]
777.         set calStimId [[[$ulPath cget -calGainObj] calIdStr]
778.         set pos [string first calibration_item $calStimId]
779.         set calStimId [string replace $calStimId $pos-1 $pos+16]
780.         set calRespId [[[$dlPathSa cget -calGainObj] calIdStr]
781.         set pos [string first calibration_item $calRespId]
782.         set calRespId [string replace $calRespId $pos-1 $pos+16]
783.
784.         setval gainSlope $gainSlope gainFlatness $gainFlatness freqList $respFreqs g
ainList $ampDiffs gainSlopeList $gainSlopeList phaseList $phaseLinList \
785.         groupDelayList $groupDelayList facAsaPwmResp $calAsaPwmResp calStim
$calStim stimPathCalId $calStimId calResp $calResp respPathCalId $calRespId\
786.         carrierFreq $carrierFreq -limitgui 20
787.
788.
789.         ## write web report charts:
790.         attachNObject -dbid $dbid -tag result -object $this -dbname "%compName" -
components {gainFlatness gainSlope \
791.         facAsaPwmResp carrierFreq calStim stimPathCalId calResp respPathCalId
dbid } -async yes
792.         attachNObject -dbid $dbid -object SCOE -async yes -components uuid
793.         attachCalibration -dbid $dbid -objects [list $ulPath $dlPathSa $dlPathPm]
794.

```



```

795.      $dbIfObj -async insertEntry -dbid $dbid -component freqList -
value $respFreqs -unit Hz -base M -dispbase M -sdesc "Response frequency" -
valuetype doublearray
796.      $dbIfObj -async insertEntry -dbid $dbid -component gainList -
value $ampDiffs -unit dB -sdesc "Gain vs frequency" -valuetype doublearray
797.      $dbIfObj -async insertEntry -dbid $dbid -component groupDelayList -
value $groupDelayList -unit s -dispbase n -base n -
sdesc "Group delay variation vs frequency" -valuetype doublearray -resolution 12
798.      $dbIfObj -async insertEntry -dbid $dbid -component phaseLinList -
value $phaseLinList -unit degree -sdesc "Phase linearity vs frequency" -
valuetype doublearray
799.      $dbIfObj -async insertEntry -dbid $dbid -component gainSlopeList -
value $gainSlopeList -unit dB/MHz -sdesc "Gain Slope vs frequency" -
valuetype doublearray
800.
801.      ## fini
802.      set logStr [niceLogString gainFlatness gainSlope]
803.      $dbIfObj -async closeResult -dbid $dbid -valid yes -
sdesc "Measurement finished: $logStr"
804.      Log r "$loggingTitle: Measurement finished ([niceTookTime -
reference $seqTime]): $logStr"
805.
806.      ## optional output of some results to file for debugging:
807.      if {$dbgf} {
808.          set fileName [getResultFileName]
809.          set pathName [file join $::SPOOL_PATH "Results" $fileName]
810.          set filep [open $pathName w]
811.          Log t "Output to file: $fileName"
812.          puts $filep "## MEAS_MCARRIER"
813.          set uuid [getScoeUUID]
814.          puts -nonewline $filep "## UUID\t$uuid"
815.          set fnsplit [split $fileName -.]
816.          puts $filep "## date-time\t[concat [lindex $fnsplit 1 ]-
[lindex $fnsplit 2]]"
817.          puts $filep [getCommonInfo]
818.          puts $filep "## Response Center\t[format %g $options(respCenter)]\tMHz"
819.          puts $filep "## Response Path\t$options(respPath)"
820.          puts $filep "## Frequency span\t$options(span)\tMHz"
821.          puts $filep "## Aperture\t$options(aperture)\tMHz"
822.          puts $filep "## Stimulus Path\t$options(stimPath)"
823.          puts $filep "## Stimulus Center\t$options(stimCenter)\tMHz"
824.          puts $filep "## Stimulus Power\t$options(stimPower)\tdBm"
825.          puts $filep "## Use calibration data\t$options(useCalSet)"
826.          puts $filep "## calAsaPwmResp\t$options(calAsaPwmResp)\tdB"
827.          puts $filep "## avgCnt\t[format %g $options(avgCnt)]"
828.          puts $filep "## atten\t$options(atten)\tdB"
829.          puts $filep "## refLevel\t$options(refLevel)\tdBm"
830.          puts $filep "## resultCount\t$options(resultCount)"
831.          puts $filep "## Evaluation center\t$options(evalCenter)\tMHz"
832.          puts $filep "## Evaluation span\t$options(evalSpan)\tMHz"
833.          puts $filep "## Gain Flatness\t[format %g $gainFlatness]\tdB"
834.          puts $filep "## Gain Slope\t[format %g $gainSlope]\tdB/MHz"
835.          puts $filep "## facAsaPwmResp\t[format %g $calAsaPwmResp]\tdB"
836.          puts $filep "## calStim\t[format %g $calStim]\tdB"
837.          puts $filep "## stimPathCalId\t$calStimId"
838.          puts $filep "## calResp\t[format %g $calResp]\tdB"
839.          puts $filep "## respPathCalId\t$calRespId"

```



```

840.         set DBID [getOption $dbid -resultCommon.id]
841.         puts $filep "## dbid\t[format %g $DBID]"
842.         puts $filep "##"
843.         puts $filep "respFreq\tgain\tgroupdelay\tphaseLin\tgainSlope"
844.         foreach f $respFreqs g $ampDiffs gd $groupDelayList pl $phaseLinList gs
      $gainSlopeList {
845.             puts $filep "[format %.2f $f]\t[format %.2f $g]\t[format %.2f $gd]\t
[format %.2f $pl]\t[format %.4f $gs]"
846.         }
847.         puts $filep "##"
848.         puts $filep "## GD abs-rel\t[CONF getval gdAbsRel]"
849.         puts $filep "## Maximum span\t[CONF getval maxSpan]"
850.         puts $filep "## Stim span expansion\t[CONF getval stimSpanExpansion]"
851.         puts $filep "## Resp span expansion\t[CONF getval respSpanExpansion]"
852.
853.         foreach stimCenter $stimCenterList {
854.             if {[info exists respFreqs$stimCenter]} {continue}
855.             puts $filep "##"
856.             puts $filep "raw data @stimCenter\t[format %.1f $stimCenter]"
857.             puts $filep "## respFreqs\trespAmpsRaw\tcalAmps\trespPhases\tcalPhas
es"
858.             foreach f [set respFreqs$stimCenter] ra [set respAms$stimCenter] ca
[set calAmps$stimCenter] rp [set respPhases$stimCenter] cp [set calPhases$stimCenter] {
859.                 puts $filep "[format %.2f $f]\t[format %.2f $ra]\t[format %.2f $
ca]\t[format %.4f $rp]\t[format %.4f $cp]"
860.             }
861.         }
862.         puts $filep "##----- END OF FILE -----## "
863.         close $filep
864.     }

```

Appendix III

```

1. ## source $TESTSEQ_PATH/rippleCompensation.tcl
2. ## define directory for returnloss measurements
3. set rldir "/home/scoe-adm/cm/uconfig3/ONEWEB-
  RFTS/ReturnlossCompensation/ReturnLoss13"
4. ## define directory for MC measurement results
5. set mcDir "/home/scoe-adm/cm/uconfig3/ONEWEB-
  RFTS/ReturnlossCompensation/Measurements13"
6. ## results will be output to desktop
7. set pi [expr {4.*atan(1)}]
8. proc cadd {x y} {
9.     ## add two complex numbers
10.    return [list [expr {[lindex $x 0] + [lindex $y 0]}] [expr {[lindex $x 1] + [lindex
  $y 1]}]]
11. }
12. proc csub {x y} {
13.     ## subtract two complex numbers
14.    return [list [expr {[lindex $x 0] - [lindex $y 0]}] [expr {[lindex $x 1] - [lindex
  $y 1]}]]
15. }
16. proc cmagphase {x} {
17.     ## convert complex number to magnitude and phase representation
18.    return [list [expr {hypot([lindex $x 0], [lindex $x 1])}] [expr {atan2([lindex $x 1
  ], [lindex $x 0])}]]
19. }
20. proc crealimag {x} {
21.     ## convert complex number to real and imaginary representation
22.    return [list [expr {[lindex $x 0] * cos([lindex $x 1])}] [expr {[lindex $x 0] * sin
  ([lindex $x 1])}]]
23. }
24. proc cmult {x y} {
25.     ## multiply two complex numbers
26.    set mag [expr {[lindex [cmagphase $x] 0] * [lindex [cmagphase $y] 0]}]
27.    set phase [expr {[lindex [cmagphase $x] 1] + [lindex [cmagphase $y] 1]}]
28.    return [crealimag [list $mag $phase]]
29. }
30. proc cdiv {x y} {
31.     ## divide two complex numbers
32.    set mag [expr {[lindex [cmagphase $x] 0] / [lindex [cmagphase $y] 0]}]
33.    set phase [expr {[lindex [cmagphase $x] 1] - [lindex [cmagphase $y] 1]}]
34.    return [crealimag [list $mag $phase]]
35. }
36. ## read in return loss of TAC input
37. set pathName [file join $rldir "TAC-input-3.csv"]
38. set filep [open $pathName r]
39. Log n "Read from file: $pathName"
40. set tacKa {}
41. while {![eof $filep]} {
42.    set line [split [gets $filep] ,]
43.    if {![isnumeric [lindex $line 0]]} {continue}
44.    set mag [lindex $line 1]
45.    set imag [lindex $line 2]
46.    lappend tacKa [list $mag $imag]
47. }
48. close $filep

```

```

49.
50. ## read in return loss of TAC output
51. set pathName [file join $rDir "TAC-output-3.csv"]
52. set filep [open $pathName r]
53. Log n "Read from file: $pathName"
54. set tacKu {}; set freqList {}
55. while {[eof $filep]} {
56.     set line [split [gets $filep] ,]
57.     if {[isnumeric [lindex $line 0]]} {continue}
58.     set mag [lindex $line 1]
59.     set imag [lindex $line 2]
60.     lappend tacKu [list $mag $imag]
61.     lappend freqList [expr {[lindex $line 0]/1e6}]
62. }
63. close $filep
64.
65. ## do it for KaUL1, KaUL2, and KaUL4 (KaUL3 is reference path for mc calibration)
66. foreach kaulNr {1 2 4} kudlNr {1 2 4} gainCorr {0 0.12 0} {
67.     ## read in return loss of KaULx cable
68.     set pathName [file join $rDir "KaUL${kaulNr}.csv"]
69.     set filep [open $pathName r]
70.     Log n "Read from file: $pathName"
71.     set kaulx {}
72.     while {[eof $filep]} {
73.         set line [split [gets $filep] ,]
74.         if {[isnumeric [lindex $line 0]]} {continue}
75.         set mag [lindex $line 1]
76.         set imag [lindex $line 2]
77.         lappend kaulx [list $mag $imag]
78.     }
79.     close $filep
80.     ## do it for KuDL1, KuDL2 and KuDL4 (KaUL3 is reference path for mc calibration)
81.     ## read in return loss of KaULx cable
82.     set pathName [file join $rDir "KuDL${kudlNr}.csv"]
83.     set filep [open $pathName r]
84.     Log n "Read from file: $pathName"
85.     set kudlx {}
86.     while {[eof $filep]} {
87.         set line [split [gets $filep] ,]
88.         if {[isnumeric [lindex $line 0]]} {continue}
89.         set mag [lindex $line 1]
90.         set imag [lindex $line 2]
91.         lappend kudlx [list $mag $imag]
92.     }
93.     close $filep
94.     ## read in return loss of KaUL3 cable
95.     set pathName [file join $rDir "KaUL3.csv"]
96.     set filep [open $pathName r]
97.     Log n "Read from file: $pathName"
98.     set kaul3 {}
99.     while {[eof $filep]} {
100.         set line [split [gets $filep] ,]
101.         if {[isnumeric [lindex $line 0]]} {continue}
102.         set mag [lindex $line 1]
103.         set imag [lindex $line 2]
104.         lappend kaul3 [list $mag $imag]
105.     }
106.     close $filep

```

```

107.      ## read in return loss of KuDL3 cable
108.      set pathName [file join $rDir "KuDL3.csv"]
109.      set filep [open $pathName r]
110.      Log n "Read from file: $pathName"
111.      set kudl3 {}
112.      while {[eof $filep]} {
113.          set line [split [gets $filep] ,]
114.          if {[!isnumeric [lindex $line 0]]} {continue}
115.          set mag [lindex $line 1]
116.          set imag [lindex $line 2]
117.          lappend kudl3 [list $mag $imag]
118.      }
119.      close $filep
120.      ## read in result KaUL3 from multicarrier measurement (reference curve)
121.      set pathName [file join $mDir "KaUL3-KuDL3.csv"]
122.      set filep [open $pathName r]
123.      Log t "Read from file: $pathName"
124.      set mcResultULDL3 {}; set freqListMeas {}
125.      while {[eof $filep]} {
126.          set line [split [gets $filep] \t]
127.          if {[!isnumeric [lindex $line 0]]} {continue}
128.          lappend mcResultULDL3 [lindex $line 1]
129.          lappend freqListMeas [lindex $line 0]
130.      }
131.      close $filep
132.      ## calculate amplitude ripple for reference path KaUL3 rippleKaUL3=20*log10(
|1 - gamma1 * gamma2|)
133.      ## gamma == S11 (real, imaginary)
134.      set rippleKaUL3 {}; set z0 [list 50 0]
135.      foreach x $stacKa y $kaul3 {
136.          lappend rippleKaUL3 [expr {20.*log10(hypot([expr {1 - [lindex [cmult $x
$y] 0]]), [lindex [cmult $x $y] 1]))}]
137.      }
138.      Log n "min rippleKaUL3=[lmin $rippleKaUL3]"
139.      Log n "max rippleKaUL3=[lmax $rippleKaUL3]"
140.
141.      ## calculate amplitude ripple for reference path KuDL3 rippleKaUL3=20*log10(
|1 - gamma1 * gamma2|)
142.      ## gamma == S11 (real, imaginary)
143.      set rippleKuDL3 {}; set z0 [list 50 0]
144.      foreach x $stacKu y $kudl3 {
145.          lappend rippleKuDL3 [expr {20.*log10(hypot([expr {1 - [lindex [cmult $x
$y] 0]]), [lindex [cmult $x $y] 1]))}]
146.      }
147.      Log n "min rippleKuDL3=[lmin $rippleKuDL3]"
148.      Log n "max rippleKuDL3=[lmax $rippleKuDL3]"
149.
150.      ## calculate amplitude ripple for path KaULx rippleKaULx=20*log10(|1 - gamma
1 * gamma2|)
151.      ## gamma == S11 (real, imaginary) is the same!
152.      set rippleKaULx {}; set z0 [list 50 0]
153.      foreach x $stacKa y $kaulx {
154.          lappend rippleKaULx [expr {20.*log10(hypot([expr {1 - [lindex [cmult $x
$y] 0]]), [lindex [cmult $x $y] 1]))}]
155.      }
156.      Log n "min rippleKaUL${kaulNr}=[lmin $rippleKaULx]"
157.      Log n "max rippleKaUL${kaulNr}=[lmax $rippleKaULx]"
158.

```

```

159.         ## calculate amplitude ripple for path KaULx rippleKaULx=20*log10(|1 - gamma
1 * gamma2|)
160.         ## gamma == S11 (real, imaginary) is the same!
161.         set rippleKuDLx {}; set z0 [list 50 0]
162.         foreach x $tacKu y $kudlx {
163.             lappend rippleKuDLx [expr {20.*log10(hypot([expr { 1 - [lindex [cmult $x
$y] 0]]), [lindex [cmult $x $y] 1]))}]
164.         }
165.         Log n "min rippleKuDL${kaulNr}=[lmin $rippleKuDLx]"
166.         Log n "max rippleKuDL${kaulNr}=[lmax $rippleKuDLx]"
167.
168.         ## read in result KaULx from multicarrier measurement
169.         set pathName [file join $mcDir "KaUL${kaulNr}-KuDL${kudlNr}.csv"]
170.         set filep [open $pathName r]
171.         Log t "Read from file: $pathName"
172.         set mcResultULDLx {}
173.         while {[eof $filep]} {
174.             set line [split [gets $filep] \t]
175.             if {[!isnumeric [lindex $line 0]]} {continue}
176.             lappend mcResultULDLx [expr {[lindex $line 1] + $gainCorr}]
177.         }
178.         close $filep
179.
180.         ## calculate theoretical corrected gain curve for KaUL1
181.         set gainCorrULDLx {}
182.         foreach gain $mcResultULDLx r1 $rippleKaULx r3 $rippleKaUL3 r2 $rippleKuDLx
r4 $rippleKuDL3 {
183.             lappend gainCorrULDLx [format %.3f [expr {$gain + $r1 - $r3 + $r2 - $r4}
]]
184.         }
185.
186.         ## output to file
187.         set desktopPath "/home/scoe-usr/Desktop"
188.         set pathName [file join $desktopPath "rippleResults3${kaulNr}.csv"]
189.         set filep [open $pathName w]
190.         puts $filep "respFreq\tgainMeasuredKaUL3\tgainMeasuredKaUL${kaulNr}\tcorrect
edGainKaUL${kaulNr}"
191.         foreach fm $freqListMeas k3 $mcResultULDL3 k1 $mcResultULDLx gc $gainCorrU
LDLx {
192.             puts $filep "$fm\t[format %.3f $k3]\t[format %.3f $k1]\t$gc"
193.         }
194.         close $filep
195.         Log n "output to file $pathName on Desktop"
196.     } ;# end foreach KaULx
197.     Log n "End of calculations!"

```