Cost effective emulation of geostationary satellite channels by means of software-defined radio

Huyen Chi Bui
Microwaves department
Télécom Bretagne-Institut Mines Télécom
Site of Toulouse (France)
Email: Huyen.Bui@telecom-bretagne.eu

Laurent Franck
Microwaves department
Télécom Bretagne-Institut Mines Télécom
Site of Toulouse (France)
Email: Laurent.Franck@telecom-bretagne.eu

***Abstract*—This contribution is about the use of software-defined radio (SDR) for channel emulation. Based on NI USRP devices and the LabVIEW programming environment, we devised an affordable RF channel emulator for geostationary satellite transmission. It is able to process 2 x 1.6 MBaud in realtime while adding thermal noise, phase noise and propagation delay according to the system which is modelled. This extensible tool is valuable for supporting teaching activities and it also serves as a convenient test bench when devising new protocols and services that are meant to run over satellite links.**

I. Introduction

Channel emulation reproduces in a lab the impairments that affect a transmission in the equivalent real world environment. Since this contribution focuses on a geostationary satellite environment, the term *channel* refers to the path from an Earth station, through a transparent (or bent-pipe) satellite and down to the destination Earth station. With that respect, it extends the meaning of channel to what is occurring inside the satellite.

Channel emulation serves different purposes: demonstration of services, performance assessment of protocols - or equipments - without the need of a full fledged transmission chain and - more importantly - without having to lease satellite capacity. Channel emulation also makes it possible to reproduce at will rare or uncontrollable events and their impact on the transmission (heavy rainfall, electromagnetic storms or strong adjacent channel interference). Hardware-based satellite channel emulators are available from the market, however with a 100+ keuro price tag, these facilities are rarely affordable to the academic or occasional customer.

Packet-level (i.e., software-based) emulators that operate at IP level offer a partial answer to the cost issue. For example, Dummynet [1] and NetEM [2], [3] are FreeBSD and Linux kernel extensions implementing mechanisms to delay, reorder, loose or corrupt IP datagrams while being processed by the network stack. Although they are useful for assessing the behaviour of protocols such as TCP, they fall short - by design -for any aspects related to RF (e.g., modem E_b/N_0 stress tests). Besides, configuring these emulators requires to abstract phenomena occurring at physical level and turn them into impairments at IP level. This task can be challenging. In [5], the authors present a packet-level emulator designed for the evaluation of transport protocol performance over satellite networks. Losses introduced by the emulator are specified in terms of Packet Error Ratio (PER), acknowledging the IP nature of the emulator. *Platine* [6] (now called *OpenSand*) is another packet-based emulator dedicated to the evaluation of DVB-RCS/S2 systems and able to emulate multiple terminals at once. As hinted by these examples, the capability to emulate satellite links is a key asset for research activities. It is also a necessary workaround against the significant cost and difficulty of getting access to real satellite resources for experimental purposes.

As far as terrestrial channel emulation is concerned, software-defined radio technology has already been used, most of the time through field programmable gate array (FPGA) platforms [7]. This contribution, as discussed later, restricts the role of FPGA to the conditioning of I/Q samples.

The present contribution discusses the design and development of a cost effective RF level channel emulator based on off-the-shelf software defined radio (SDR) hardware. The advent of affordable SDR platforms such as the Universal Software Radio Peripheral (USRP) brings to users powerful technological blocks.

As far as emulation is concerned, the advantages of SDR are two-fold. First, the solution proposed here is cost effective (about 8 keuro). Second, it is highly extensible/customisable, most of the signal processing being done on a desktop computer. While the limits of the approach are dictated by the available computer power and computer-to-SDR interface speed, our experience shows that a forward and return traffic of 1.6 Mbaud QPSK 3/4 (i.e., 2.4 Mbit/s at IP level in each direction) can be handled in real time by a single desktop computer. Performance can be doubled with two computers. These capabilities cover common satellite usage scenarios where consumer bi-directional data traffic is involved. It is also fit for the occasional emulation user or in an academic context.

Section II presents the architecture and design of the emulator. A combination of National Instruments' USRP hardware and LabVIEW software is used. Section III discusses the measurements that were conducted to validate the behaviour of the emulator and the limitations we discovered. Lessons learned are also hi lighted. Section IV shows the emulator in action on a sample case. Finally, Section V concludes and hints to further development directions that look promising.

II. Architecture and Design

Figure 1 shows the system that is considered for emulation. It is a common scheme for Internet access with forward and return channels. The customer premises are equipped with a very-small aperture terminal (VSAT) comprising a modem, a block up converter (BUC), a low noise amplifier block (LNB) and a dish antenna. On the operator site, similar components are used bearing in mind that some of them are scaled up so to serve multiple customers at once. The customer and operator sites communicate through a geostationary telecommunication satellite. We consider here that the transponders are transparent and Ku-band. Multiple carriers co-exist in a single transponder.

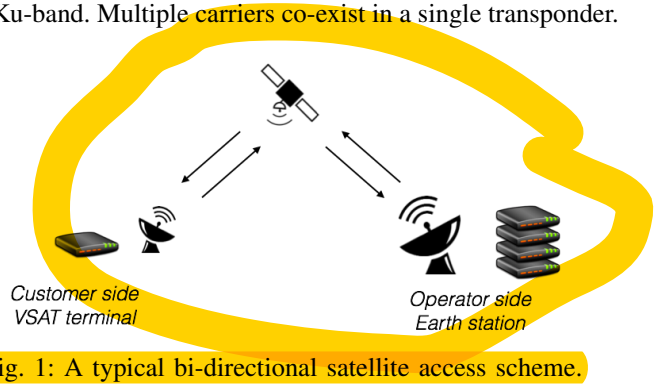

Fig. 1: A typical bi-directional satellite access scheme.

A. Emulation hardware

In order to emulate such a system, all the elements that reside between the two modems are removed. The emulator mimics their impact on the signal so to "fool" the two modems and let them believe that the original components are still in place. One advantage of this approach is to work with RF signals that display low frequencies (ca. 1 GHz) and low power (ca. -35 dBm) since the high-power amplifiers and up-converters are not physically present in the emulator.

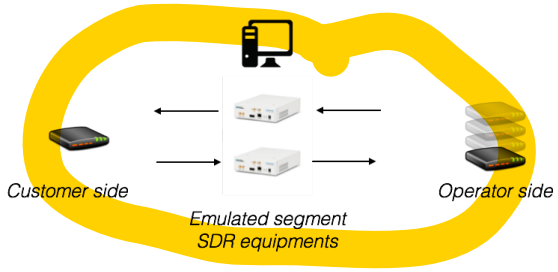

Fig. 2: An emulated bi-directional satellite access scheme. Modems and SDR devices are connected by means of coaxial cables.

The emulated system is presented in Figure 2. The two modems support the DVB/S2 standard and are configured to work in a single channel per carrier (SCPC) mode. The signals on the forward and return links are analog-to-digital (A/D) converted by two Universal Software Radio Peripheral (USRP) devices and the resulting I/Q samples are decimated and sent to a PC running National Instrument's LabVIEW. Table I lists some key parameters and characteristics that are used in the following description.

TABLE I: Key characteristics and parameters of the set up.

Modems	
Make and model	Newtec EL-470
Modem RX/TX frequency	L-band (ca. 1-2 GHz)
Carrier modulation	QPSK
Carrier symbol rate	256 kbaud - 1.6 Mbaud
SDR devices	
Make and model	2 x National Instruments USRP 2920
Frequency coverage	50 MHz-2.2 GHz
Noise figure	5-7 dB
Maximum bandwidth at 16-bit sampling	20 MHz
Signal sampling rate	4 * the carrier symbol rate
Host communication	1 Gb/s Ethernet with VITA-49 encapsulation
Host PC	
Processor	3.6 GHz Xeon quad core
RAM	6 GB
Operating system	Windows 7 - 64 bit
SDR software	NI LabVIEW 13.0 - 32 bit

From now on, we will focus on a single channel (i.e., either the forward or return channel) since the system is fully symmetrical as far as RF processing is concerned. The signal is transmitted from the modem and fed to the USRP RX port. It is then down-converted by a RF front-end, A/D converted and then decimated to the requested sample rate. The USRP range of SDR equipment being entry-level, no further processing is implemented in the FPGA and the resulting I/Q samples are transmitted through a gigabit Ethernet link to a computer for software processing. Once done, the modified I/Q samples are sent back to the USRP for D/A conversion, up-conversion (i.e., from baseband to L band) and transmission to the destination modem. The next paragraphs describe the processing which is done in LabVIEW within the computer.

B. Emulation software

The emulation software is composed of three parallel processes: the RX loop, the emulation loop and the TX loop. Considering that the signal is processed in realtime, care is taken regarding the timing constraints. The RX and TX loops handle the communication from/to the USRP devices so to fetch/provide I/Q samples at the right rate. Samples are conveyed to/from the USRP in groups called "frames".

The emulation loop is the most complex of the three and subject to customisation depending on what phenomena are implemented in the emulator. The list of phenomena currently supported are: (a) signal power gain as a result of antenna gain, Earth station and satellite amplification, (b) signal attenuation as a result of free space loss, weather conditions and other marginal losses occurring in the equipments, (c) thermal noise as a result of earth and sky brightness and electronic noise, (d) intermodulation products and interferences, (e) phase noise generated by electronic devices and (f) propagation delay resulting from the travel distance. The four first factors can be expressed as variation of carrier power to noise power ratio (C/N). The I/Q samples received from the USRP are then impaired accordingly. Next, phase noise expressed as noise power spectral density at a certain offset from the main carrier is added. Finally, the I/Q samples are delayed so to match the requirements (typically 250 ms).

Most of these factors are currently static. For example, the emulator provides a parameter for defining the weather on the

uplink of the forward channel (clear sky, cloudy, light rain, heavy rain). These choices correspond to typical attenuation values (over South West of France) that are hardcoded in the emulator. While this approach may seem simplistic, all the mechanisms are now in place to support dynamic attenuation based on - say - time series of rain intensity. The same goes for the phase noise: a future evolution is to integrate phase noise templates as provided by the DVB-S2 standard [4]. Another possibility is the inclusion of models such as Saleh's [8] for assessing the amount of phase noise depending on the operating point of the satellite high-power amplifier.

C. User interface

Figure 3 represents the interface of the forward link composed of the following control/indicator areas (from top to bottom): system definition, SNR and timing, USRP control (bottom left), signal visualisation (middle) and debugging (right).

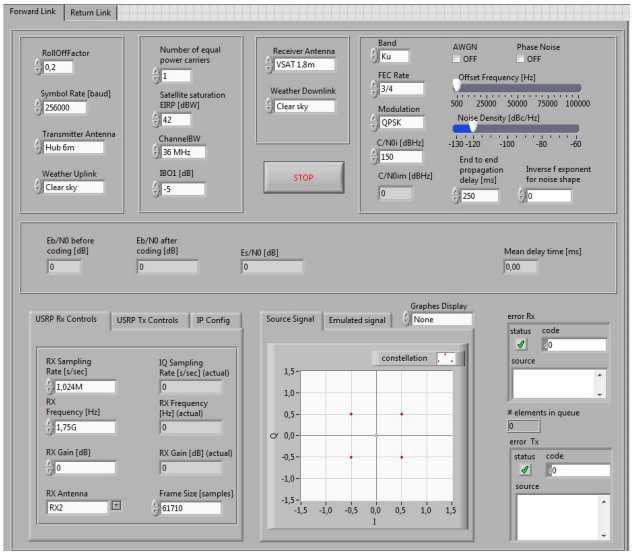

Fig. 3: The user interface with from top to bottom : system definition, SNR and timing, USRP controls (left), signal visualisation (middle) and debugging (right).

The first area is reserved to system parameters settings. The user provides information about the carrier characteristics (filter roll-off, symbol rate, modulation, code rate), the RX and TX receiver antennas (1.2m, 1.8m, 3m, 6m, 10m), the weather conditions on the uplink and the downlink (clear sky, cloudy, light rain, heavy rain), transponder parameters (input back off (IBO), number of equal power carriers, satellite saturation point, bandwidth) and other parameters (signal to interference ratio). The second area shows the SNR (expressed as a ratio between the energy per bit and the noise power spectral density, E_b/N_0) resulting from the above configuration. The time taken by a sample to travel from the input to the output of the emulator is also displayed. The third area contains configuration values for the USRP (e.g., IP address, operating frequency, sampling rate). The fourth area - signal visualisation - displays on an I/Q diagram the signal constellation before/after adding impairments. It provides helpful hints about how the different system parameters impact the quality of the modulated signal. The last area is used for debugging purposes.

III. Verifications, Limitations and Lessons Learned

A. Verifications

The first verification is of system scope. The target E_S/N_0 computed by the emulator and used to impair the signal was compared to the E_S/N_0 estimated by the receiving modem. In all cases, we found a difference at most equal to 0.10 dB.

The accuracy of the signal timing was also verified by comparing two configurations: one where modems are directly connected back-to-back and one with the emulator in between. Doing so made it possible to measure round trip delays and check that the delays actually generated by the emulator are correct with respect to what is requested through the configuration. It also permitted to determine that the time required for processing the signal in the emulator software is around 10 ms depending on the impairments that are implemented and other parameters (e.g., sample rate, number of samples per acquisition frame). In addition, the delay required for going through the emulator software and hardware is 60 ms. This - comparably - large delay still fits within the 250 ms enveloppe observed during a typical hop through a geostationary satellite.

Finally, error vector magnitude (EVM) measurements were conducted to assess the impact of the emulator hardware on the signal quality. For that purpose, the signal transmitted from the modem (QPSK 3/4 at 2.0 Mbaud) was compared to the signal transmitted from the USRP when no impairments are implemented. In this case, the USRP operates as a transparent RF relay by means of a simple LabVIEW program that forwards I/Q samples from the RX to the TX port. Table II shows the relevant metrics (Annex 1 shows all the measurements).

TABLE II: EVM measurements comparing the modem and USRP outputs for a QPSK modulation at 2.0 Mbaud, received power of -30 dBm

EVM metric	Modem	USRP
Mean EVM (RMS, %)	0.92	1.82
Stdev EVM (RMS, %)	0.05	0.18
Mean magnitude error (RMS, %)	0.43	0.71
Stdev magnitude error (RMS, %)	0.01	0.12
Mean phase error (RMS, deg)	0.46	0.96
Stdev phase error (RMS, deg)	0.03	0.10
Mean carrier frequency error (Hz)	-401.01	-831.50
Stdev carrier frequency error (Hz)	3.05	9.01

Without surprise, the inexpensive USRP hardware introduces its own impairments. Additional measurements with the above transmission parameters show that the E_S/N_0 degrades from 28.30 dB to 27.80 dB. These are not typical E_S/N_0 values found with satellite communications, however it should be recalled that no impairments are introduced in the transmission.

B. Current limitations

An important characteristic of the emulator is the streaming operation that requires strict timing. The duration for processing each iteration of the RX and TX loops should be equal. Conversely, the emulation loop is required to be - at least - as fast as the others (i.e., processing one samples frame in the

emulation loop must be at least as fast as receiving/transmitting a samples frame).

This upper bound is approximated through the ratio of the frame size (number of samples) over the I/Q rate. Assuming the duration of one emulation iteration is longer than the frame duration, the TX loop will wait for data to transmit and the TX buffer feeding the USRP RF front end will eventually underflow. Therefore, each I/Q rate calls for the configuration of a corresponding frame size complying to these timing rules.

The second limitation concerns the timing accuracy of the USRP clock. When the I/Q rate is increased (e.g., to 1 Msample/s), the USRP onboard clocks are not stable enough resulting in the DVB modems loosing the carrier lock. This phenomena occurs even if the emulated E_S/N_0 level is well above the demodulation threshold. The solution is to drive the USRP by means of an external clock (often found on measurement instruments that can export their 10 MHz clock signal) or a GPS disciplined oscillator.

C. *Lessons learned*

The lessons learned are summarised in the following 5-fold wisdom: know your science, know your SDR device, know your drivers, know your operating system and know your programming environment.

Before elaborating, it is worth mentioning that our usage of the SDR can be nicknamed "high performance SDR" in a similar way to "high performance computing". With that respect, the limitations listed before as well as the lessons learned are biased and should not be considered as extensively representative of other SDR uses.

What we learned is that developing SDR-based channel emulation is as challenging as it is rewarding. Because of the real-time nature of processing, all the components that make up the processing chain are important. A typical example is the transport driver (Ethernet in the case of the USRP). As far as real time streaming of I/Q samples is concerned, not all network interface cards are made equal and their associated driver requires tuning. Building a multi-disciplinary team to handle SDR projects seem to be the right approach.

IV. Experimenting with the emulator

This Section shows how the emulator can be used for assessing the impact of transmission conditions on the performance of TCP. TCP is well known for being significantly sensitive to large bandwidth-delay product channels and error-prone links as found in geostationary satellite environments. In all experiments, we use *Iperf* to transfer a 2.5 MB file from one modem to the other. Two versions of TCP are used: *Reno* and *Cubic*. The carrier is modulated with QPSK 3/4 at a symbol rate of 1.024 Mbaud using normal (i.e., 64800 bit) FEC frames. The resulting round trip time (measured by means of *ping*) is ca. 800 ms (500 ms of RTT and 300 ms of various packetisation delays). Measurements are collected using *tcpdump* and analysed with *wireshark*.

Figure 4 shows the throughput over time for TCP *Cubic* when the E_S/N_0 is set to 4.0 dB (left) and 3.9 dB (right). For the former, the goodput (i.e., user data volume over total transmission duration) is 111 kB/s while the later displays a goodput of 6.6 kB/s. Keeping in mind that the threshold for quasi-error free demodulation is at 4.03 dB, the surge of the bit error rate at 3.9 dB caused 543 kB of data retransmission (about one fifth of the original data volume). In addition to that, the TCP dynamics for throughput adjustment in the presence of errors are quite harmful.

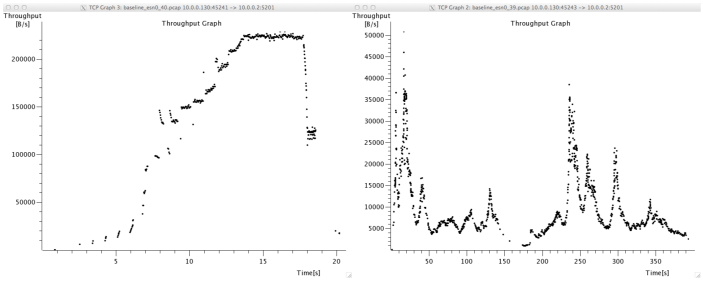

Fig. 4: Comparison of TCP *Cubic* throughput when the $E_S/N0$ at the receiver is set to 4.0 dB (top) or 3.9 dB (bottom).

The second experiment shown in Figure 5 compares the throughput of TCP *Cubic* (left) and *Reno* (right). In both cases, the E_S/N_0 is set to 4.0 dB. For *Cubic*, the measured goodput is 111 kB/s while for *Reno* it is 39.5 kB/s. *Cubic* demonstrates how its improved slow start law is better suited to the satellite environment. Indeed, during the whole experiment, *Reno* is trapped in a slow start phase demonstrated by the ever growing throughput until the transfer is terminated. In comparison, *Cubic* ends slow start after 12 seconds to reach a plateau.

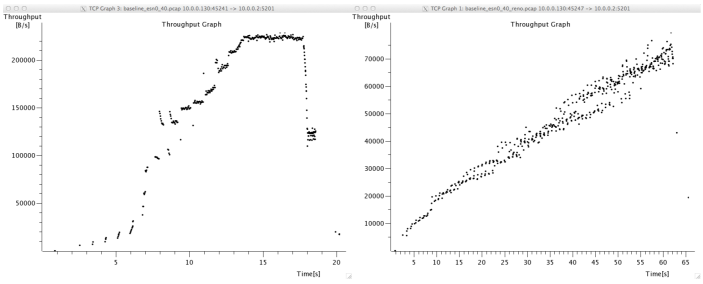

Fig. 5: Comparison of TCP throughput for *Cubic* (left) and *Reno* (right) at an $E_S/N0$ of 4.0 dB.

V. Conclusions and perspectives

In this work we demonstrated how to use inexpensive software defined radio devices in order to build a channel emulator that features performance fit for the academic and occasional user. Our solution is able to process in realtime up to twice 1.6 Mbaud of QPSK modulated signals. As of now, the emulator is used for supporting students' projects and research activities.

Work is still ongoing with several objectives. The first one is the optimisation of the signal processing in the emulator so to speed it up. Pipelining techniques are used to exploit the inherent loop parallelism of LabVIEW. At the time of writing these lines and thanks to this optimisation, the processing capability has been increased by 25 %, that is 2 Mbaud (both

on the forward and return links) of processing in real-time. The second objective is the addition of new features in the emulator such as rain fading based on time series and improved phase noise modelling.

We are also porting the emulator platform to more powerful SDR devices that include the ability to customise processing in the FPGA. Our goal is to assess the tradeoff that can be achieved trying to balance cost, complexity and performance. As a preliminary result, Annex 2 shows EVM measurements for the NI VST PXI 5644R when the equipment is used as a transparent RF relay (similarly to measurements conducted with the USRP in Section III).

ACKNOWLEDGMENT

This work is funded through the French inter-ministerial programme, grant number F1310008M called *STAMP*. The authors are also grateful to support received from National Instruments.

REFERENCES

[1] L.Rizzo, *Dummynet: a simple approach to the evaluation of network protocols*, ACM SIGCOMM Computer Communication Review 27 (1), 31-41, 1997.

[2] L. Nussbaum, O. Richard, *A Comparative Study of Network Link Emulators*, Proceedings of the 12th Communications and Networking Simulation Symposium (CNS'09), March, 2009, San Diego, USA.

[3] S. Hemminger, *Network Emulation with NetEm*, Proceedings of the 6th Australia's National Linux Conference (LCA2005), April 2005, Canberra, Australia.

[4] *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, EN 302 307, European Telecommunications Standards Institute (ETSI).

[5] Caini, C.; Firrincieli, R.; Lacamera, D., "SAT01-5: An Emulation Approach for the Evaluation of Enhanced Transport Protocols Performance in Satellite Networks," Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE , vol., no., pp.1,5, Nov. 27 2006-Dec. 1 2006.

[6] Baudoin, C.; Arnal, F., "Overview of Platine emulation testbed and its utilization to support DVB-RCS/S2 evolutions," Advanced satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), 2010 5th , vol., no., pp.286,293, 13-15 Sept. 2010.

[7] Mar, J.; Chi-Cheng Kuo; You-Rong Lin; Ti-Han Lung, "Design of Software-Defined Radio Channel Simulator for Wireless Communications: Case Study With DSRC and UWB Channels," Instrumentation and Measurement, IEEE Transactions on , vol.58, no.8, pp.2755,2766, Aug. 2009.

[8] A. Saleh, *Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers*, IEEE Transactions on Communications , vol. COM-29, no. 11, pp. 1715-1720, Nov. 1981.

ANNEX 1 : EVM MEASUREMENTS FOR MODEM AND USRP

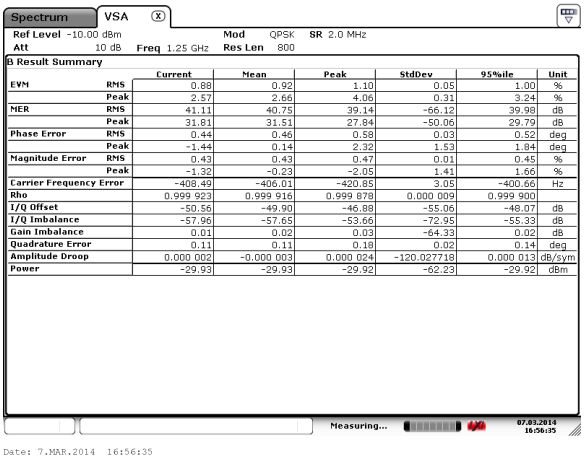

Fig. 6: EVM measurement when modems are set up "back to back".

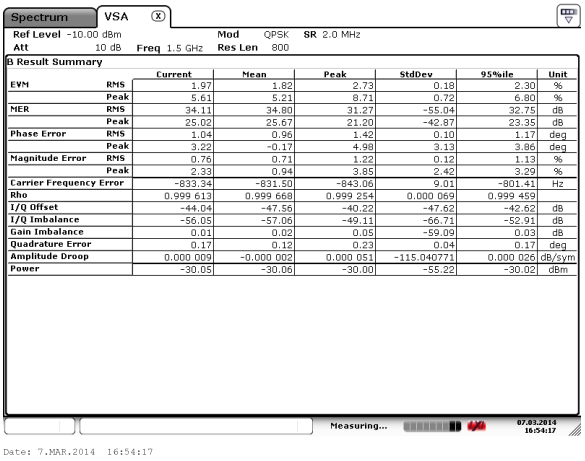

Fig. 7: EVM measurement with NI USRP 2920 used as a transparent relay.

ANNEX 2 : EVM MEASUREMENTS FOR NI VST

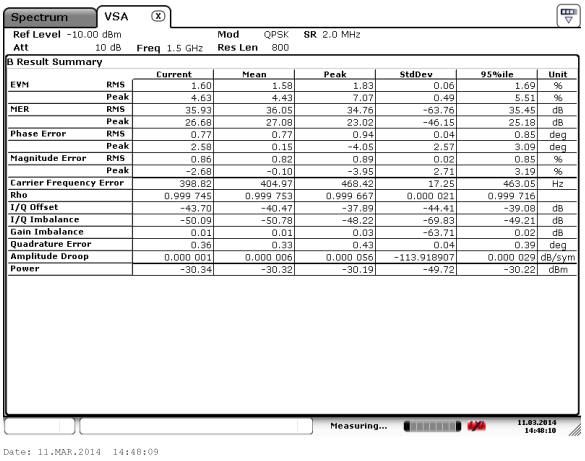

Fig. 8: EVM measurement with NI VST PXI 5644R used as a transparent relay.