Off-the-Shelf Reconfigurable Software Define Radio Approach for Vector Network

Mohd Nazrin Mohd Yassin[1]; Shaiful Jahari Hashim[2]; Zubaida Yusoff [3]
[1,2] Dept. of Computer and Communications Systems, Universiti Putra Malaysia, 43400, Selangor, Malaysia
[3]Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
[1]nazrin@gmi.edu.my; [2]sjh@upm.edu.my; [3]zubaida@mmu.edu.my

Abstract - **The goal of this work is to develop a promising low cost vector network analyzer measurement system based on reconfigurable Software Define Radio (SDR) using minimal components and utilizing open source signal processing framework, GNU Radio. This work was motivated by the possibility to realize a low cost and a fast development cycle time by eliminate the SoC system building. The traditional method utilizing the GNU Radio relies on its GUIs only is not flexible, rigid and unable to acquire the baseband data immediately. A new methodology to perform data acquisition from the GNU Radio software is developed to add more flexibility on signal characteristic from SDRs. The developed methodology comprises of Python scripts that are automatically generated and open source packages from GitHub Linux project. As a conclusion, the RF characterization on the Device under Test (DUT) to determine S-parameter of reflection coefficient can be measure quickly on SDR platform.**

Index Terms - **Software Define Radio, Open-Source Software, RF Network Characterization, S-parameters measurement**

1. Introduction

Network analyzers are dedicated instruments that have been developed to characterize Radio Frequency (RF) devices as a function of frequency (Mubarak et al., 2018; Marimuthu et al., 2016). In general, network analysis creates data model of return loss, gain or impedance characteristics of a linear network through stimulus-response over the frequency range of interest. These characteristics that are known as S-parameters can be determined by measuring the phase and magnitude of the RF waves. The measurement is a key component of many engineering application especially microwave frequency range which quantify data in terms of real and imaginary terms. This key component can be used to understand the construction of proposed vector network analyzer based on hardware and commercial SDR. Although many SDR platforms are specific for wireless communications (Li & Huang, 2017; Dang et al., 2017; Uengtrakul & Bunnjaweht, 2014), they can be used for RF measurement as well and become the dominant technology.

Today the SDR systems have been designed around field programmable gate arrays (FPGAs) on larger SoC system which required FPGA programming using Hardware Description Language such as Verilog or VHDL (Rigoni et al., 2018). Therefore time-to-market for product development may delay because a researcher requires a lot of learning involved in developing signal processing functions on the FPGA hardware. For example, the FPGA programmer may need to design a quadrature signal modulator or a signal generator from the design entry level using hardware description language. However, in recent years, RF measurement research has benefited from the development of the open-source GNU Radio software which enables non-hardware experts to build custom software radios that can be used with a low-cost software-defined radio (SDR) device. At the same time, the SDRs are not only low cost but advantages in small size, portable and require small power to operate.

In the following section, we start by reviewing the targeted SDR hardware selection and the software overview. All requirements for open source software are described. Then the discussion is followed by the drawback of GNU Radio software when representing the response of RF signal in its GUI. A new technique on replacing RF signal on GUI with a graph that can represent RF signal characterization is described. On these activities, we can make a conclusion on the SDR performance from RF characterization response. We continue this work by implementing the S-parameters measurements using the SDR hardware and the GNU Radio software without extensive studies in programming. Standard calibration process is also described and verified with a graphical view on the magnitude and phase on the Smith chart. Then, a comparison between the characteristic of the reflection coefficient on a sample device under test (DUT) between a commercial VNA and the proposed system will be discussed.

The main contributions from this work can be summarized as follows:

- The development of the SDR that is coupled with SoC Zynq System without FPGA programming skills or without building Soc system from the scratch.

- The implementation of SDR based transceiver for measuring scattering parameter (s-parameters) using a fully open source environment, namely VNA-SDR.
- The measurement results on the reflection coefficient from a sample DUT using the developed system is accurately comparable with commercial industrial VNA.

2. SDR Hardware and Software Overview

In this paper, we used a Xilinx Zynq System On-Chips (SOC) based platform known as Zedboard carrier board, coupled with an Analog Devices RF front end, FMCOMMS2 daughter board as shown in Fig. 1. This daughter board consists of AD9361, an integrated RF agile transceivers chip which contains two receivers and two transmitters available for RF signals path. The chip operates between 70 MHz to 6 GHz frequency range, and supports channel bandwidths from 200 kHz to 56 MHz (Shi et al., 2015). The AD9361 is based on direct-conversion receiver architecture, therefore the proposed measurement system is based on the homodyne principle that only have a single local oscillator (LO). This LO provides the stimulus signal and is also used to process the response. Many literatures have explained the internal block diagram of Zynq SoC (Xilink, 2018; Moorthy & Kapre, 2015) and AD9361 (Zhao & Yao, 2016; Harikrishnan et al., 2014).

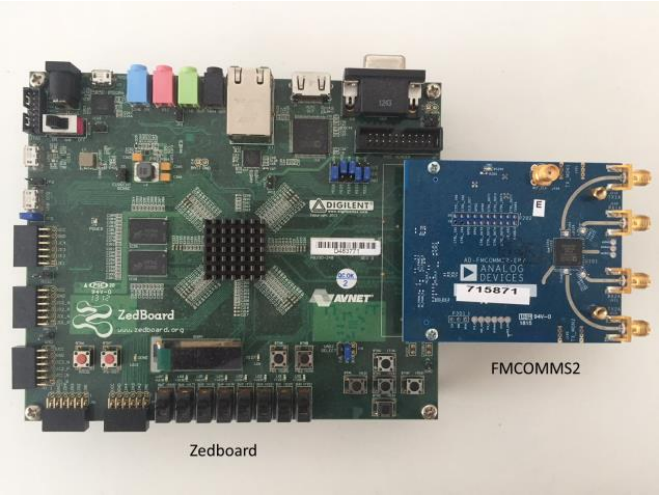


Fig. 1: ZedBoard is a Low-Cost Development Board for the Xilinx Zynq-7000 SoC and FMCOMMS2 is an FPGA Mezzanine Card (FMC) Board for the AD9361, a Highly Integrated RF Agile Transceiver

Signal acquisition is carried out in GNU Radio Companion (GRC) as a software framework. GNU Radio is a collection of clever signal processing modules that allow general computers to perform real-time software defined radio and other functions with relatively simple and inexpensive hardware peripherals. GNU Radio provides a variety of displays for different signal domains representation, such as time, frequency, number, and constellation sink. All these functions and signal presentation can work without hardware implementation (Gandhiraj et al., 2012). However in this work, the physical RF wave signals transmitted and received are defined by the GNU Radio and are implemented on FMCOMMS2 board. A python programming language as a firmware was developed to perform user input interface, post processing, import predefines packages such as numpy, scipy and scikit-rf and finally to plot the multiple graphs using matplotlib package. This firmware reads both binary formats of real and imaginary data to convert into touchstone file format as shown in Fig. 2. The open source package dedicated for RF/Microware engineering knows as scikif-rf (Arsenovic et al., 2013; Arsenovic et al., 2018) used for network analysis and calibration with no additional cost. This library provides various data plot to display magnitude, phase, complex plane, and smith chart.

In order for the open source software GNU Radio is able to communicate to hardware boards, there are three files must be downloaded and configured from Analog Devices repositories, https://github.com/ analog devices inc. The files are the libiio library files used for interfacing with Linux Industrial Input/Output (IIO) Subsystem. The Linux IIO subsystem is to provide support for devices such as the analog to digital or the digital to analog converters (ADCs, DACs), the Direct Digital Synthesis (DDS), the Phase Locked Loops (PLLs) and the RF transceivers. This library also acts as a server for real-time data acquisition and has the control ability on a remote computer or embedded form. Due to the high data rate and the CPU usage during processing baseband signals, the work focuses on computer-hosted form. Another file required is gr-iio file used for IIO blocks in GNU Radio, and libad9361-iio file used for IIO AD9361 library.

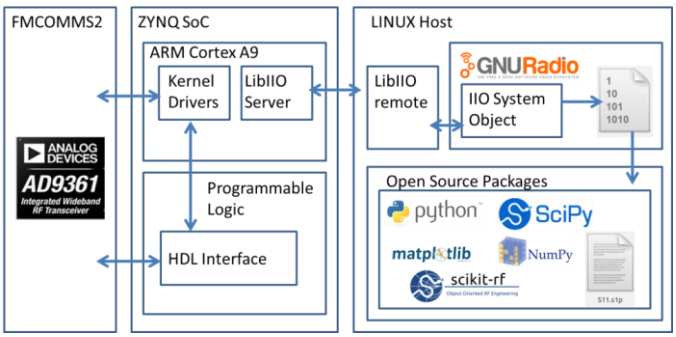


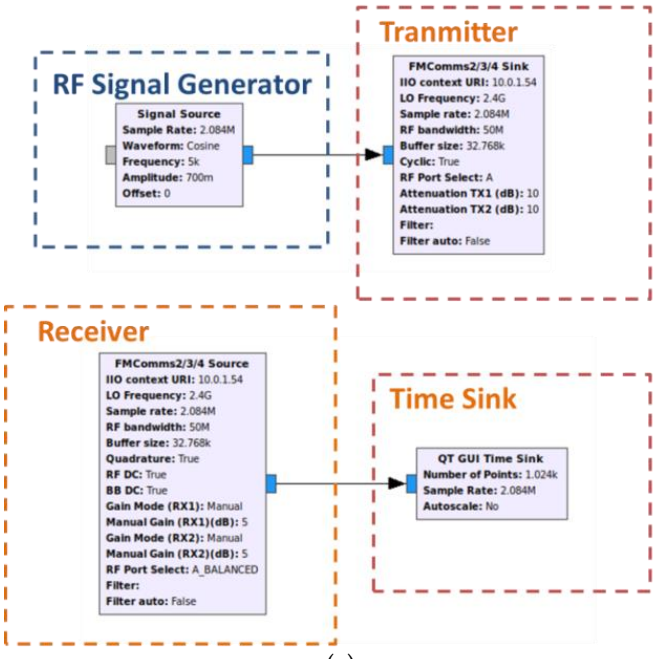
Fig. 2: Firmware Architecture

In addition, the FMCOMMS2 board comes with the FPGA software images, so we can focus on implementing the functionality of the radios module on the remote computer side. Software radios are built using the Python programming language in GNU Radio, by connecting signal processing components together into a flowgraph, the inputs and the outputs of which are connected to the SDR via a libiio driver interface.

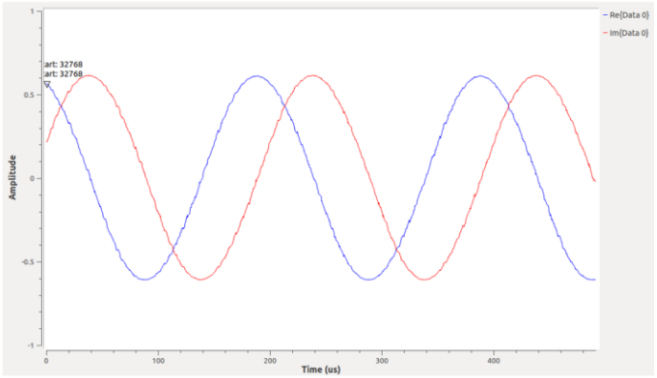
3. Effective GNU Radio Data Acquisition

Specifically, the Fig. 3(a) shows a simplify graphical representation of this sequence's flowgraph in the GNU Radio Companion (GRC), a GUI-based flowgraph editor packaged with a GNU Radio. A GNU Radio flowgraph is made up of the RF signal generation, the SDR blocks that consist of transmitter-receiver, and the time sink blocks, which are connected by the virtual wires that transmit the baseband signals between them.

The RF signal generator passes the 5kHz stimulus signal to the transmitter SDR block, FMCOMMS2 Sink with the minimum sampling rate of 2.084 MS/s. The FMCOMMS2 Sink block is physically interfaced to the transmit channels of the FMCOMMS2 SDR. Then the demodulated received signal comes back into the flowgraph via the FMCOMMS2 Source block which interfaces to the receive channels of the FMCOMMS2 SDR. The Time Sink block is added at end of the chain to measure the magnitude of the receiving signal wave when the flowgraph is executed as shown in Fig. 3(b).



(a)



(b)

Fig. 3: (a) A Simplify Version of Magnitude RF Signal Measurement from GNU Radio Companion Blocks (b) The Time Sink Graphical View of Receiving Wave on Receiver Part of SDR.4

This example illustrates that GNU Radio flowgraphs run continuously and are not recording data for any analysis. Furthermore, this type of GUI of GNU Radio lacks on the ability to perform sweep on any parameters, which is crucial for network analyser. The measurement system requires the frequency being swept through the measurement band. Another drawback of this method is that the user is unable to analyse and collect the magnitude and phase of that signals directly from the graph. Therefore, the characterization on this SDR as a measurement system is limited.

To overcome this problem, a new method is developed without using GUI on GNU Radio but utilizing multiple open source python packages that is powerful in numerical data computation and flexible on data plotting. A improve version of magnitude measurement on GNR Radio flowgraph is shown in Fig. 4 with added blocks. The RF Signal generator, the transmitter and the receiver is equipped with complex-valued data type. Then a complex to real block and a complex to imaginary block are added on the last chain and are stored using File Sink block as DAT file format. All signals constructed in GNU Radio are baseband signal and RF Signal from/ to transceiver must be in the quadrature I and Q data format. No GUI is selected in 'generate' option during the execution of the flowgraph on GNU Radio, but this option is replaced with the binary file format that will be used by open source packages like numpy and scipy. The fresh GNU Radio application is automatically generated, namely top_block.py where this application can generally create a gr_top_block, instantiate the blocks, connect the blocks together, and then start the gr_top_block. Then, this flowgraphs are configured by adding additional library of open source packages as shown in Fig. 5 such as numpy, scipy and matplotlib.

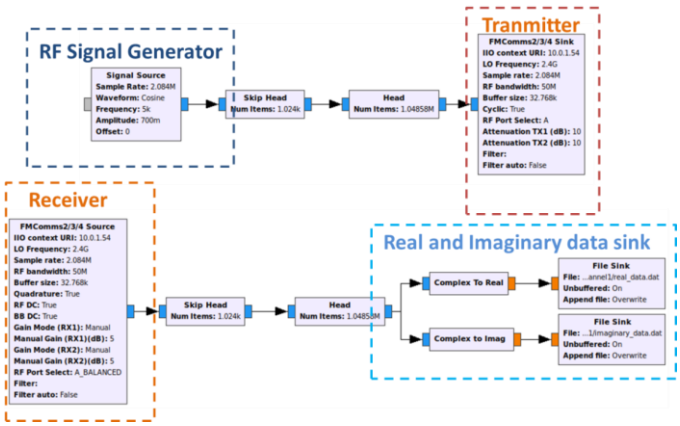


Fig. 4: GRU Radio to Generate Baseband Signal and Stored a Binary File Format

top_block.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Top Block
# Generated: Fri Mar 23 17:43:32 2018
##################################################

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import gr
from gnuradio import iio
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser
import numpy
import scipy
import time
import matplotlib.pyplot as plt
from gnuradio import channels
import random
import skrf as rf
import pylab
from datetime import datetime
from numpy import inf
from scipy.interpolate import spline
from math import factorial
import os
import sys
import math
```

modules from the gnuradio package

Import another open source packages

Fig. 5: Python file allows adding modules or packages and can be reused efficiently

The next block is a class called the 'top_block', which consist of all the instantiate and functions to add the blocks and how to connect them. Variables or parameters are defined to control sampling rate, gain, frequency, local oscillator (LO) and amplitude of the signal generators. Storage folders that contain the real and the imaginary data are also define as shown in Fig. 6.

Finally, the user application can be added into the main function that is available at the bottom of the python script as shown in Fig. 7. In this work, an open source python library, scipy is used to read and manipulate both the real and the imaginary binary data for readable magnitude value. Meanwhile matplolib is used as a plotting library that produce quality figures in a variety of hardcopy formats.

```
class top_block(gr.top_block):

    def __init__(self, Gain=5, LO=2400000000, RF_bandwidth=50000000, amplitude=0.7,
 frequency=5000, sample_rate=2084000):
        gr.top_block.__init__(self, "Top Block")

        ##################################################
        # Parameters
        ##################################################
        self.Gain = Gain
        self.LO = LO
        self.RF_bandwidth = RF_bandwidth
        self.amplitude = amplitude
        self.frequency = frequency
        self.sample_rate = sample_rate

        ##################################################
        # Blocks
        ##################################################
        self.iio_fmcomms2_source_0 = iio.fmcomms2_source_f32c('10.0.1.54', LO, sample_rate,
 RF_bandwidth, True, False, 0x8000, True, True, True, "manual", Gain, "manual", Gain,
"A_BALANCED", '', False)
        self.iio_fmcomms2_sink_0 = iio.fmcomms2_sink_f32c('10.0.1.54', LO, sample_rate,
 RF_bandwidth, True, False, 0x8000, True, "A", 10.0, 10.0, '', False)
        self.blocks_skiphead_0_0 = blocks.skiphead(gr.sizeof_gr_complex*1, 1024)
        self.blocks_skiphead_0 = blocks.skiphead(gr.sizeof_gr_complex*1, 1024)
        self.blocks_head_1 = blocks.head(gr.sizeof_gr_complex*1, 1024*1024)
        self.blocks_head_0 = blocks.head(gr.sizeof_gr_complex*1, 1024*1024)
        self.blocks_file_sink_0_0 = blocks.file_sink(gr.sizeof_float*1,
'/home/nazrin/mygnuradio/fmcomms2/channel1/imaginary_data.dat', False)
        self.blocks_file_sink_0_0.set_unbuffered(True)
        self.blocks_file_sink_0 = blocks.file_sink(gr.sizeof_float*1,
'/home/nazrin/mygnuradio/fmcomms2/channel1/real_data.dat', False)
        self.blocks_file_sink_0.set_unbuffered(True)
```

Controlled parameters

Blocks

Fig. 6: GNU Radio Generates Python Template for Instantiate Blocks and Calling Functions

```
def main(top_block_cls=top_block, options=None):
    if options is None:
        options, _ = argument_parser().parse_args()

    tb = top_block_cls(Gain=options.Gain, LO=options.LO, RF_bandwidth=options.RF_bandwidth,
amplitude=options.amplitude, frequency=options.frequency, sample_rate=options.sample_rate)
    tb.start()
    tb.wait()


if __name__ == '__main__':
    main()
```

Write user application

Fig. 7: An User Application in the Main Function for Data Extraction, Manipulate and Plotting a Graph

As a conclusion, a python programming is applied mainly to extract the real and the imaginary data produced by GNU Radio and to reuse in developing a network analysis application in the following sections.

4. VNA-SDR System Architecture

This work aims to minimize a lot on the cost and complexity of the measurement hardware. Furthermore, the signal post-processing and the display tools are applied by using scikit-rf open source at no additional cost. In this section, we describe the proposed measurement system and its critical components of measuring S-parameters, which integrates an RF component, some cables and the DUT. The selected DUT is one port passive LTE dipole antenna. The topology structure described in this paper is aimed for S11 only to cover the scope of study.

The measurement system consists of a software part, where the signal acquisitions or computations are implemented, and a hardware part that is responsible for the up and down conversion of the RF waveform. Block diagram of the proposed measurement system is shown in Fig. 8. It contains four main components:

- A remote computer used to do the data acquisition, the error correction and to display

various format analysis such as the magnitude response, the phase and the Smith chart.
- Two set of receivers and transmitters that are used as reference channel and test channel respectively.
- Directional couplers set that separates the reflected waves at the test channel and incident waves at the reference channel.
- A passive one-port RF component as the DUT.

In this paper, we showed a simple transmission/reflection (T/R) network analyzer test. This test used one fixed source to port one to measure forward S-parameters. This measurement system only measures the forward direction, S11 and S21. However to measure the reverse parameters like the S12 and the S22, the DUT needs to be disconnected and physically reversed and then the measurement will be repeated. In this paper, only the S11 measurement is presented as a basic test for the passive device.

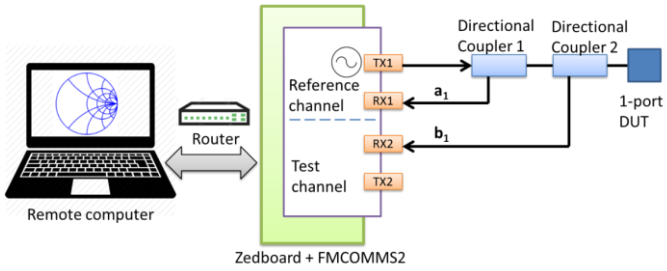


Fig. 8: One-Port Transmission & Reflection Measurement System Architecture

4.1 Complex Divider Implementation

As shown in Fig. 8, the signal generated is passed through the directional couplers into the one-port DUT. Then the signal from the DUT will reflect some fraction of its reflection amplitude wave, b_1, from the incident wave amplitude, a_1. The directional coupler 1 serves to separate and measure the incident wave amplitude, a_1 through receiver RX1. Meanwhile the reflection wave amplitude, b_1 is fed into receiver RX2. The DUT input reflection coefficient S_{11} can be calculated from these measured wave amplitudes.

$$S11 = \frac{b1}{a1} \tag{1}$$

$$S11{\cdot}real + jS11{\cdot}imaginary = \frac{b1{\cdot}real + jb1{\cdot}imaginary}{a1{\cdot}real + ja1{\cdot}imaginary} \tag{2}$$

$$S11{\cdot}real = \frac{(a1{\cdot}real)(b1{\cdot}real) + (a1{\cdot}imaginary)(b1{\cdot}imaginary)}{(a1{\cdot}real)^2 + (a1{\cdot}imaginary)^2} \tag{3}$$

$$S11{\cdot}imaginary = \frac{(a1{\cdot}real)(b1{\cdot}imaginary) - (a1{\cdot}imaginary)(b1{\cdot}real)}{(a1{\cdot}real)^2 + (a1{\cdot}imaginary)^2} \tag{4}$$

This S_{11} is a complex number consisting of a real and an imaginary part. In order to calculate S_{11} in the signal processing block, a complex divider block is developed in GRC as shown in Fig. 9. The result of this operation, S_{11} real and S_{11} imaginary values will be stored into File Sink block in terms of binary file format.

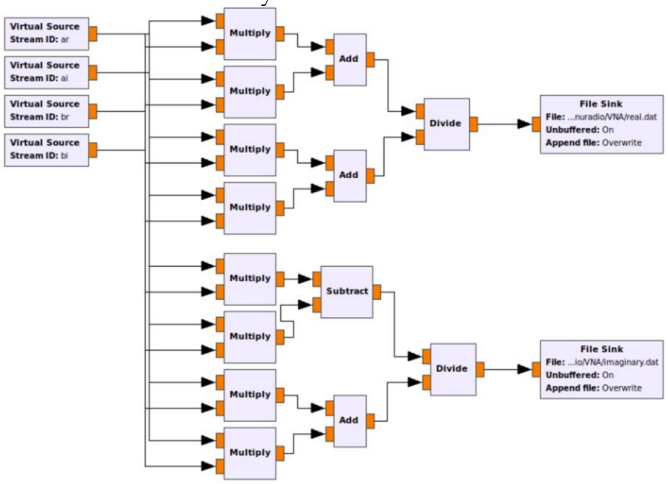


Fig. 9: A complex divider block implementation.

An overview of the transceiver structure as exposed to the GRC is depicted in Fig. 10. The FMCOMMS2 Sink block acts as the transmitter, which generates a sine signal from the Signal Source block and the FMCOMMS2 Source block deploys two receivers RX1 and RX2.

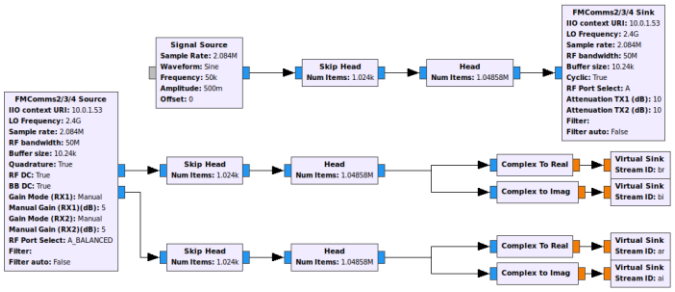


Fig. 10: GRC View for Transceiver Blocks

4.2 S11 Measurement Topology

In general the signal source or the Continuous Wave (CW) produces the incident signal used to stimulate the DUT. Then the DUT responds by reflecting part of the incident energy and transmitting the remaining part. By sweeping the excitation frequency, the frequency response of the DUT can be determined. A test set consists of the directional element is required for separating the incident wave on the DUT and the reflected wave from the DUT. Fig. 11 shows the hardware configuration on measuring S11 for one port DUT such as antenna.

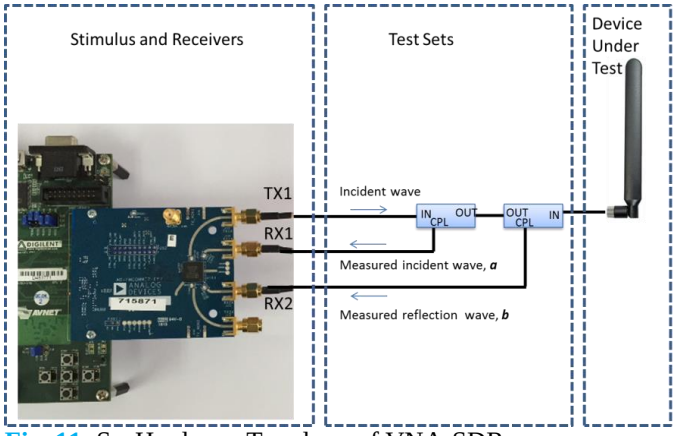


Fig. 11: S_{11} Hardware Topology of VNA-SDR

4.3 One-port Calibration using Short-Open-Load Technique

In this section we present a basic calibration technique using Short-Open-Load (SOL) set standard kit. This VNA-SDR measures the phase and magnitude relationship of incident and reflected waves with well-known calibration standard attached to the reference plane as shown in Fig. 12. In the SOL technique, the three standards are measured to determine the one port model error known as 'Open', 'Short' and 'Load' (Ridler & Rumiantsev, 2008)..

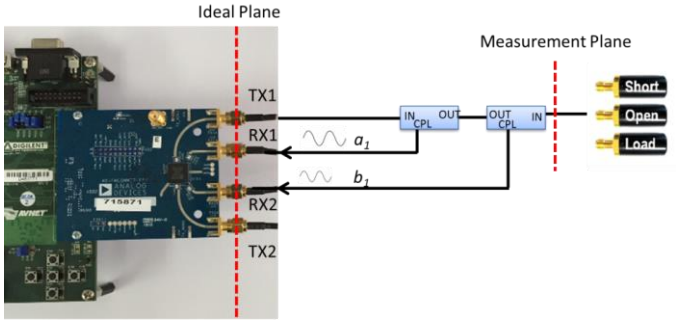


Fig. 12: VNA-SDR with Known Calibration Standard

For simplification, only 'Open' standard measurement is described in this section. Fig. 13 shows the measurement of the 'Open' standard response on Smith chart view. Without the calibration, the 'Short' describes a spiral circle near the edge of Smith chart. After the calibration is applied, the phase rotation has been removed and the 'Open' shows up as point on the right side of the Smith chart, with all frequencies being measured falling on the same point as shown in Fig. 14. Fig.15 shows how one-port calibration compensate initial reflection coefficient from -6 dB to 0 dB after removing all measurement errors.

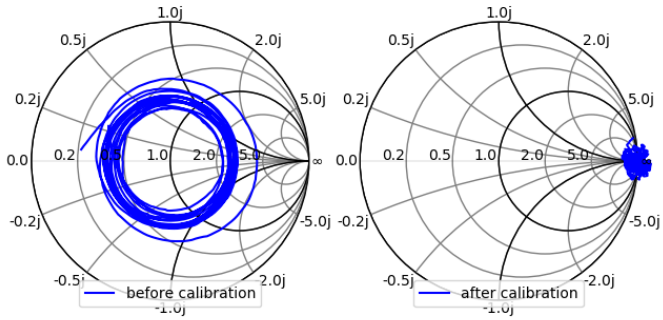


Fig. 13: 'Open' Measurement Standard before and after Calibration on Smith Chart

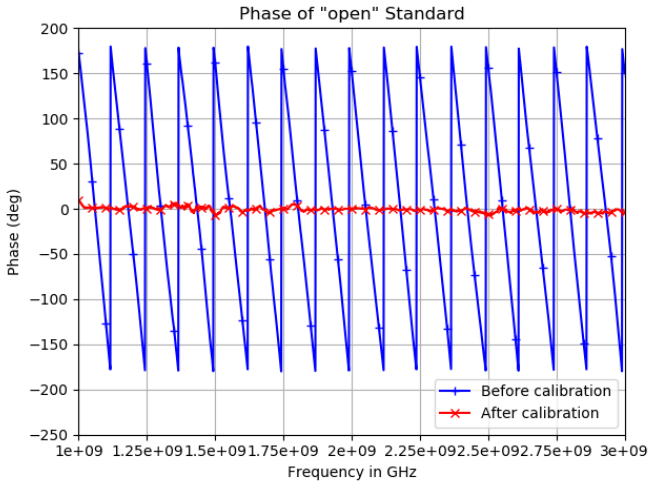


Fig. 14: Phase after Calibration

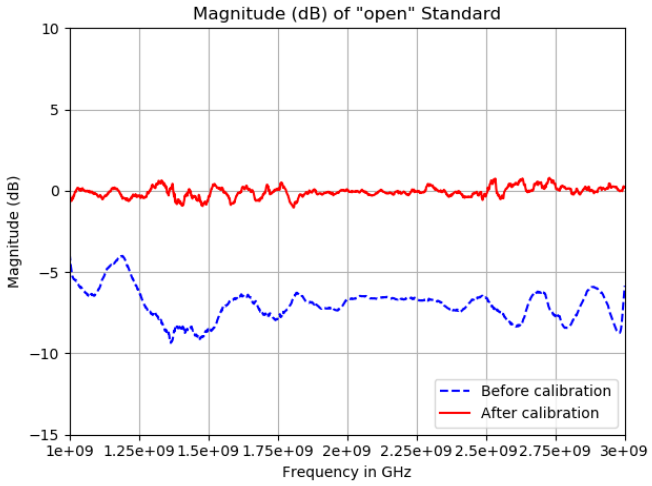


Fig. 15: Improvement of Magnitude 'Open' Measurement after Calibration

4.4 S11 for Passive DUT

The LTE Dipole Antenna from Pulse Series SPDA 24700/2700 is a DUT for the one-port test of this work. Fig. 16 shows a high precision comparison of the reflection coefficients between the VNA-SDR and the Keysight E5071C measurements on a passive LTE Dipole

Antenna in the fundamental frequency range up from 1 GHz to 3 GHz.

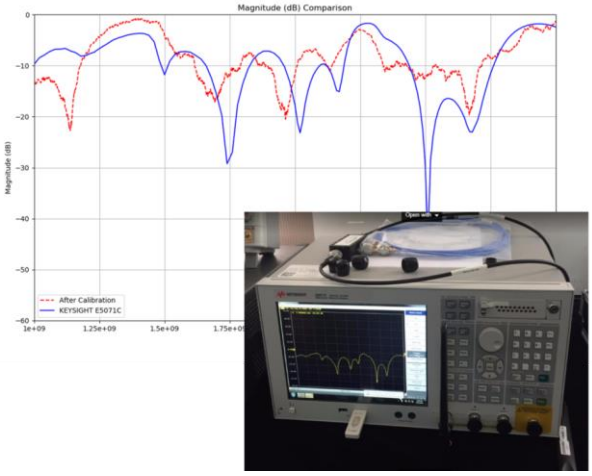


Fig. 16: Comparison of S11 Measurement on a commercial VNA Instrument and the VNA-SDR

5. Conclusions

We presented a new approach for wideband frequency VNA-SDR scattering-parameter measurements S11 utilizing a software defined radio and an open source signal processing software. This technique also contributes on reducing the system complexity and the component count, resulting in a smaller test set at a lower cost, and less programming skills required, which makes it attractive for small measurement.

A number of experiments and verification has been carried out on the LTE dipole antenna and the results show a reliable performance as compared to a standard calibration and reference specification by manufacturer (Pulse Electronics, 2018). Finally we have demonstrated the proposed measurement system that is inexpensive and relatively simple to realize in an open source environment. The proposed system utilizes a reconfigurable hardware and a software to measure reflection coefficient or S11 of a dipole antenna as the selected DUT. This measurement system is also easily to be deployed for forward coefficient S21 measurement on two-port devices such as amplifiers, and filters.

References

Arsenovic, A., Chen, L., Bauwens, M. F., Li, H., Barker, N. S., and Weikle, R. M. (2013). An Experimental Technique for Calibration Uncertainty Analysis. In IEEE Transactions on Microwave Theory and Techniques, vol. 61, no. 1, pp. 263-269.

Arsenovic et al. RF and Microwave Engineering Scikit [Online] Cited 2018-05-21. Available at: https://github.com/scikit-rf/scikit-rf

Dang, T. M., Gonnot, T. and Saniie, J. (2017). End-to-end wireless digital communication system of FPGA based software defined radio. IEEE International Conference on Electro Information Technology (EIT), pp. 235-239.

Gandhiraj, R., Ram, R. and Soman, K.P (2012). Analog and Digital Modulation Toolkit for Software Defined Radio. Procedia Engineering, Volume 30, 2012, Pages 1155-1162.

Harikrishnan, B., Raghul, R. , Shibu, R. M. and Nair, K. R. (2014). All programmable SoC based standalone SDR platform for researchers and academia, First International Conference on Computational Systems and Communications (ICCSC), Trivandrum, pp. 384-386.

Li, Y. and Huang, X. (2017). High speed communication and realization between FPGA and DSP in software-defined radio system. International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 2329-2332.

Marimuthu, J., Bialkowski, K. S. and Abbosh, A. M. (2016). Software-Defined Radar for Medical Imaging. In IEEE Transactions on Microwave Theory and Techniques, vol. 64, no. 2, pp. 643-652.

Moorthy, P. and Kapre, N. (2015). Zedwulf: Power-Performance Tradeoffs of a 32-Node Zynq SoC Cluster. IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines, pp. 68-75.

Mubarak, F. A. and Rietveld, G. (2018). Uncertainty Evaluation of Calibrated Vector Network Analyzers. In IEEE Transactions on Microwave Theory and Techniques, vol. 66, no. 2, pp. 1108-1120.

Pulse Electronics, Larsen Antennas. LTE Dipole. 4 pages. [Online] Cited 2018-06-22. Available at: https://productfinder.pulseeng.com/product/SPDA24700/2700

Ridler, N. and Rumiantsev, A. (2008). VNA Calibration. IEEE Microwave Magazine, vol. 9, No 3, pp 86-99.

Rigoni, A., Manduchi, G. , Luchetta, A. , Taliercio, C. and Schröder, T. (2018). A framework for the integration of the development process of Linux FPGA System on Chip devices. Fusion Engineering and Design, vol. 128, pp. 122-125.

Shi, T., Guo, W., Yang, L., and Li, A. (2015). Remote wideband data acquiring system based on ZC706 and AD9361, IEEE International Wireless Symposium (IWS 2015), Shenzhen, pp. 1-4.

Uengtrakul, B. and Bunnjaweht, D. (2014). A cost efficient software defined radio receiver for demonstrating concepts in communication and signal processing using Python and RTL-SDR. Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), pp. 394-399.

Xilinx Inc. Zynq-7000 All Programmable SoC Data Sheet. 25 pages. [Online] Cited 2018-06-22. Available at: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

Zhao, C. and Yao, X. (2016). A digital hardware platform for RF PA digital predistortion algorithms. 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1133-1137.