TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JOOSE TAMMINEN

DIGITAL CONTROL OF RF SELF-INTERFERENCE CANCELLER IN FULL-DUPLEX RADIO

Master of Science thesis

Examiner: Prof. Mikko Valkama
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 6th of April 2016

ABSTRACT

JOOSE TAMMINEN: Digital Control of RF Self-Interference Canceller in Full-Duplex Radio
Tampere University of Technology
Master of Science thesis, 56 pages, 5 Appendix pages
August 2016
Master's Degree Program in Electrical Engineering
Major: Embedded systems
Examiner: Prof. Mikko Valkama
Keywords: Full-duplex, cancellation, FPGA

In traditional wireless communication systems, transmission and reception are divided in either time or frequency domain. In-band full-duplex means that the transmission and reception take place on the same frequency simultaneously, theoretically doubling the spectral efficiency. The most significant challenge in wireless full-duplex communication is the self-interference, which causes the systems own transmission signal to be coupled into the receiver. An analog canceller is designed to remove this self-interference from the reception signal. The cancellation takes place entirely in the RF domain.

Any variation in the surroundings of the antenna also affect the self-interference. A control system is required to track these changes and adjust the canceller accordingly. This thesis presents a digital control system for the canceller. The control system is implemented using a field-programmable gate array (FPGA).

The canceller and the control system were developed at Tampere University of Technology (TUT) in collaboration with Intel Labs. The project was concluded in January 2016, when the finished setup was delivered to Intel Labs.

Using the digital control system, the canceller is capable of canceling up to 68 dB, 66 dB and 63 dB of the self-interference from the reception signal with 20 MHz, 40 MHz and 80 MHz signal bandwidths respectively. Roughly 20 dB of the cancellation originates from the intrinsic attenuation between the transmitter and the receiver. The control system is also capable of reacting and adapting to any changes in the self-interference quickly in order to maintain sufficient cancellation in a dynamic environment.

TIIVISTELMÄ

JOOSE TAMMINEN: Itseishäiriön Kumoajan Digitaalinen Ohjaus
Tampereen Teknillinen Yliopisto
Diplomityö, 56 sivua, 5 liitesivua
Elokuu 2016
Sähkötekniikan diplomi-insinöörin koulutusohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: Prof. Mikko Valkama
Avainsanat: Full-duplex, kumoaminen, FPGA

Langattomassa tiedonsiirrossa lähetys ja vastaanotto ovat tyypillisesti jaettu joko aika- tai taajuustasossa. Full-duplex tarkoittaa, että lähetys ja vastaanotto tapahtuvat samalla taajuudella samanaikaisesti, teoreettisesti tuplaten spektritehokkuuden. Keskeisimpänä haasteena langattomassa full-duplex tiedonsiirrossa on itseishäiriö, jossa järjestelmän oma lähetyssignaali kytkeytyy vastaanottimeen. Analoginen kumoaja on suunniteltu poistamaan itseishäiriö vastaanottosignaalista. Kumoaminen tapahtuu täysin RF taajuudella.

Muutokset antennin ympäristössä vaikuttavat myös itseishäiriöön. Kumoajan ohjausjärjestelmä seuraa itseishäiriön muutoksia ja säätää kumoajaa niiden mukaisesti. Tässä diplomityössä esitellään digitaalisen ohjausjärjestelmän toteutus käyttäen FPGA-piiriä (engl. ’field-programmable gate array’).

Kumoaja ja ohjausjärjestelmä kehitettiin Tampereen Teknillisellä Yliopistolla yhteistyössä Intel Labsin kanssa. Projekti saatettiin päätökseen tammikuussa 2016, kun viimeistelty laitteisto toimitettiin Intel Labsille.

Käyttäen digitaalista ohjausjärjestelmää, kumoaja kykenee kumoamaan jopa 68 dB, 66 dB ja 63 dB itseishäiriöstä vastaanotetusta signaalista 20 MHz, 40 MHz and 80 MHz kaistanleveyksillä vastaavasti. Tästä noin 20 dB on peräisin lähettimen ja vastaanottimen välisestä ominaisesta vaimennuksesta. Ohjausjärjestelmä kykenee myös reagoimaan ja sopeutumaan itseishäiriön muutoksiin nopeasti säilyttäen tarpeeksi tehokkaan kumoamisen dynaamisessa ympäristössä.

PREFACE

The work presented in this thesis began in June 2015 and was concluded in January 2016. The writing process began at the end of 2015, once the design was finalized and the results were obtained.

I would like to thank professor Mikko Valkama who gave me the opportunity to work on this full-duplex research project. This project allowed me not only to apply the knowledge obtained during my studies but also learn a lot of new things beyond the scope of embedded systems.

I would also like to give my thanks to my colleagues Matias Turunen, Dani Korpi and Enrico Manuzzato, whom I've had the pleasure to work with on this project. Thanks to Yang-Seok Choi and Timo Huusari from Intel. I also greatly appreciate the time and effort of the people who gave me feedback on this thesis.

Finally, I would like to express my gratitude to my family and friends who have supported me throughout my studies.

Tampere, August 2, 2016
Joose Tamminen

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROGRAMS

LIST OF ABBREVIATIONS AND TERMS

ABBREVIATIONS

ALM	**adaptive logic module**
ADC	**analog-to-digital converter**
AGC	**automatic gain control**
BPF	**band-pass filter**
DAC	**digital-to-analog converter**
DLS	**dithered linear search**
DDR	**double data rate**
ENOB	**effective number of bits**
EMI	**electromagnetic interference**
FPGA	**field-programmable gate array**
FIR	**finite impulse response**
FIFO	**first-in-first-out**
FDD	**frequency domain duplexing**
IIR	**infinite impulse response**
IQ	**in-phase and quadrature**
I/O	**input/output**
ISR	**interrupt service routine**
LSB	**least significant bit**
LMS	**least mean squares**
LO	**local oscillator**
LVDS	**low-voltage differential signaling**
LNA	**low-noise amplifier**
LPF	**low-pass filter**
MSB	**most significant bit**
OCT	**on-chip termination**
PAPR	**peak-to-average power**
PLL	**phase-locked loop**
PA	**power amplifier**
PCB	**printed circuit board**
RF	**radio frequency**
RSSI	**received signal strength indicator**

RX	**reception**
SI	**self-interference**
SPI	**serial peripheral interface**
SINAD	**signal-to-noise-and-distortion ratio**
SNR	**signal-to-noise ratio**
SoI	**signal-of-interest**
SDR	**single data rate**
SMA	**sub-miniature type-a**
TDD	**time domain duplexing**
TX	**transmission**
USB	**universal serial bus**
UART	**universal asynchronous receiver transmitter**
UI	**user interface**
VGA	**variable-gain amplifier**
VST	**vector signal transceiver**
VHDL	**very high speed integrated circuit hardware description language**
WLAN	**wireless local area network**

TERMS

Altium Designer	PCB schematic and layout design software.
Balanced	Balanced signals are carried differentially over two near-identical conductors. This provides excellent common-mode rejection.
Balun	A component that converts balanced signal between balanced and single-ended.
MATLAB	Matrix laboratory, a programming language widely used in numerical computing.
ModelSim	VHDL and Verilog simulation environment by Mentor Graphics.
Quartus II	Programmable logic development environment by Altera.
RJ45	An 8 position 8 contact modular connector.
Single-ended	Single-ended signals use a single conductor that is typically referred to ground e.g. coaxial cable.

1. INTRODUCTION

Over the past decade the amount of mobile devices has increased tremendously. It is common that the data is no longer stored physically on the device, but is instead accessed when needed through cloud services. Additionally, the popularity of audio and video streaming services has increased the data traffic significantly. This lead to a rapid development of faster wireless communication systems. However, the current methods of wireless communications are quickly approaching theoretical limits.

All radio systems require a certain amount of bandwidth in the electromagnetic spectrum. The use of the radio spectrum is highly regulated, often by governments. A significant portion of the radio spectrum has been allocated for purposes other than wireless mobile communications. Therefore, it is important to make the most efficient use of the limited spectrum available. Spectral efficiency indicates the rate at which information can be communicated over a given bandwidth.

In traditional wireless communication systems transmission and reception are separated in either time or frequency, known as *time domain duplexing* (TDD) and *frequency domain duplexing* (FDD) respectively. However, it is obvious that a system capable of transmitting and receiving simultaneously on the same frequency would double the spectral efficiency. This is referred to as in-band full-duplex.

In wired systems full-duplex operation is easily achieved using separate cables. In wireless systems the medium, through which the signal propagates, is shared for both transmission and reception. A significant obstacle for in-band full-duplex systems is *self-interference* (SI), where the systems own transmission signal overpowers the received signal. Removing the SI is necessary for a viable in-band full-duplex operation. In the analog domain the SI can be reduced either by increasing the isolation between the transmitter and the receiver or using an active canceller. The canceller presented in this thesis uses the transmission signal to create a complementary signal, which cancels the SI.

For a long time, the simultaneous transmission and reception on the same frequency was considered impossible. As a result of recent research advances, that stance no longer applies. In the past few years, multiple methods for SI cancellation have been developed. These methods include analog *radio frequency* (RF) cancellation [1, 2, 3] as well as digital cancellation [4].

The main contribution of this thesis is a digital control system, which was designed and built for third revision of a specific analog RF canceller architecture [1, 5]. Both the canceller and the control system were developed in collaboration with Intel Labs. Two functioning copies of the devices were built. The control system comprises of an *analog-to-digital converter* (ADC), a *digital-to-analog converter* (DAC) and a *field-programmable gate array* (FPGA). An adaptive filter algorithm is utilized to adjust the cancellation signal of the canceller such that the SI is minimized from the received signal. Additionally, the control system must be able to maintain good cancellation in a dynamic environment, where the surroundings of the antenna vary. The digital control system was also presented in a conference paper [6] and the cancellation results were included in a magazine article [7].

This thesis is structured as follows. The second chapter covers the theoretical aspect of in-band full-duplex operation and provides the background for RF cancellation and the control system. The third chapter focuses on the detailed technical implementation of the control system. The fourth chapter explains the measurement setup and results, which is followed by comparison to other results presented in the academia. The final chapter is the conclusion of this thesis and it also provides a viewpoint on how the control system could be developed further.

2. THEORY AND BACKGROUND

This chapter provides the motivation and requirements for analog RF cancellation and explains the causes of the SI. The architecture of the analog RF canceller is also explained at a general level. Finally, the theoretical background of the control algorithm is described briefly, before analyzing the different aspects of control methods.

The analog RF canceller presented in this chapter is a third revision of the design, which is based on a paper by Yang-Seok Choi and Hooman Shirani-Mehr titled "Simultaneous transmission and reception: Algorithm, design and system level performance" [5]. The previous two revisions of this canceller architecture are presented in a master's thesis by Timo Huusari titled "Analog RF Cancellation of Self Interference in Full-Duplex Transceivers" [8]. The third revision of the canceller introduces a third tap and the analog control system is replaced with a digital one.

2.1 In-band full-duplex

Duplexing refers to different methods of achieving bidirectional communication over a shared medium. In RF communication two common methods are FDD and TDD. In FDD the transmission and reception signals are divided into two different frequency channels. TDD only uses a single frequency channel, but it can only transmit or receive at a given time. *Wireless local area network* (WLAN) is a typical example where TDD is utilized. FDD and TDD are half-duplex methods because the transmission and reception do not happen simultaneously on the same channel. In-band full-duplex, hereafter referred to as full-duplex, uses only a single channel to transmit and receive at the same time. The differences between the three duplexing methods are illustrated in Figure 2.1. Full-duplex would theoretically double the spectral efficiency. Compared to FDD the required frequency bands would be halved and compared to TDD the capacity of the specific frequency channel would be doubled.

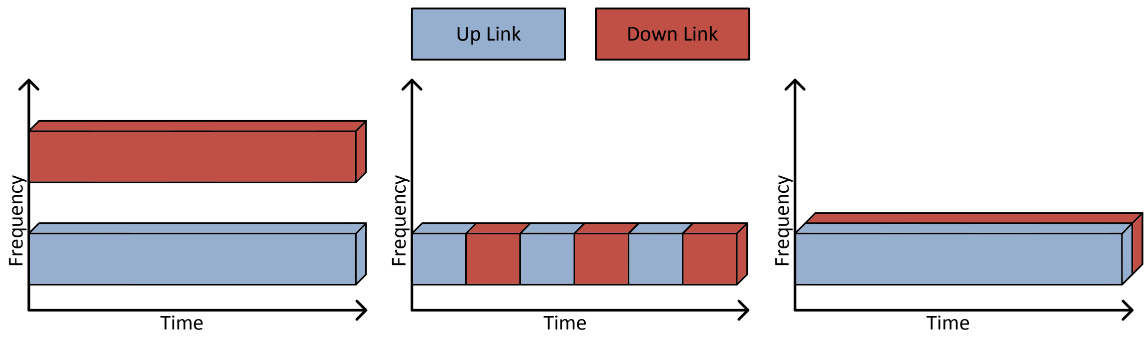


Figure 2.1 *From left to right: frequency domain duplexing, time domain duplexing and in-band full-duplex.*

The downside of full-duplex is the SI, unwanted coupling of systems own *transmission* (TX) signal to the *reception* (RX) signal. The SI is the result of non-ideal isolation between the transmitter and receiver. Typical TX signal can be over 90 dB more powerful than the *signal-of-interest* (SoI), which contains the data to be received. Even though the SI is attenuated intrinsically, there is a huge difference remaining in the signal powers. The SoI is lost under SI from the TX signal. In order to make the SoI signal detectable again, the SI must be reduced significantly. Simply put, TX signal is subtracted from the RX signal leaving behind the SoI. The SI cancellation is done in both analog and digital domains. The purpose of analog cancellation is to reduce the power of the SI such that the dynamic range of the receiver can cover both the SI and the SoI. The remaining SI is canceled digitally [4].

A common transceiver architecture widely used in mobile devices is the direct conversion transceiver, where the baseband signal is upconverted directly to the carrier frequency determined by the *local oscillator* (LO). A high-level block diagram of a direct conversion transceiver is depicted in Figure 2.2. The figure also shows where the digital and analog cancellation take place.

In the transmitter chain, the complex data is converted into analog baseband signals using a DAC. The baseband *in-phase and quadrature* (IQ) signals are then filtered using a *low-pass filter* (LPF) before upconverting them to the carrier frequency using an IQ mixer. The RF signal is then amplified to the required power using a *variable-gain amplifier* (VGA) and a *power amplifier* (PA).

In the receiver chain the received RF signal is first filtered using a *band-pass filter* (BPF). A *low-noise amplifier* (LNA) amplifies the low-power RF signal, before it is downconverted into baseband IQ signals. The signals are then filtered using a LPF. To utilize the

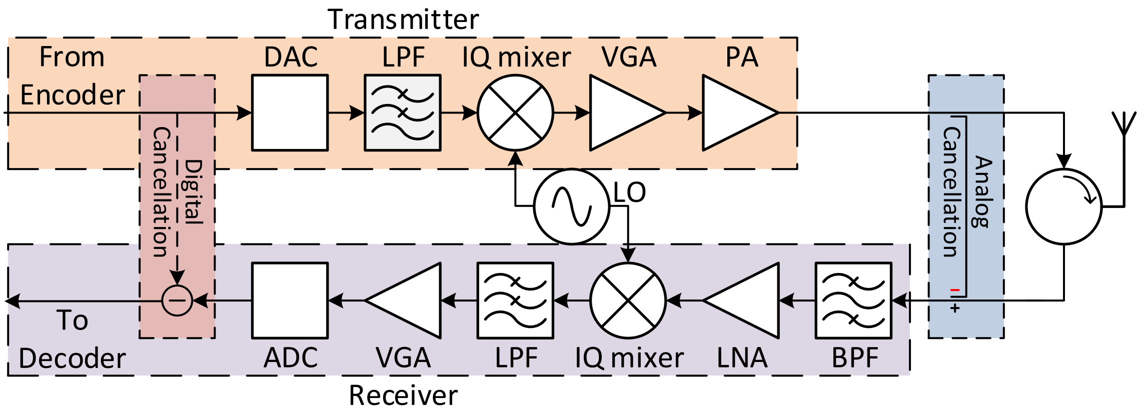


Figure 2.2 *Block diagram of a full-duplex direct conversion transceiver using a circulator.*

full dynamic range of the receiver ADC the signals are amplified using a VGA such that the input power of the ADC remains constant.

Consider a generic WLAN scenario. The transmit power is limited to 100 mW, which corresponds to 20 dBm. The signal bandwidth is 20 MHz. Assuming a 10-bit receiver ADC, the dynamic range will be approximately 60 dB. In order to prevent the RX signal from saturating the receiver, a minimum of 60 dB of analog cancellation is required. The power before the ADC has to be brought down further than the saturation limit to account for the *peak-to-average power* (PAPR) of the signal. The analog cancellation comprises of intrinsic attenuation, resulting from isolation between the transmitter and receiver, and active cancellation provided by the canceller. Illustration of the WLAN example is provided in Figure 2.3. The power levels presented in the figure are referenced to the thermal noise floor at the LNA input.

The thermal noise floor P_N is determined by the room temperature T and the signal bandwidth Δf

$$P_N = 10 \times log_{10}(kT\Delta f \times 1000), \tag{2.1}$$

where k is the Boltzmann's constant in joules per kelvin and the unit of the power is dBm. In standard room temperature of 300 K this results in thermal noise floor of -101 dBm, -98 dBm and -95 dBm for 20 MHz, 40 MHz and 80 MHz bandwidths respectively. In reality all receivers add their own noise to the system, according to the receiver's noise figure. Lower noise figure value indicates better performance. In the WLAN example the noise figure is assumed to be around 10 dB.

Signal-to-noise ratio (SNR) determines the theoretical upper limit for maximum data rate with any given bandwidth according to the Shannon-Hartley theorem [9]. If the SI is not

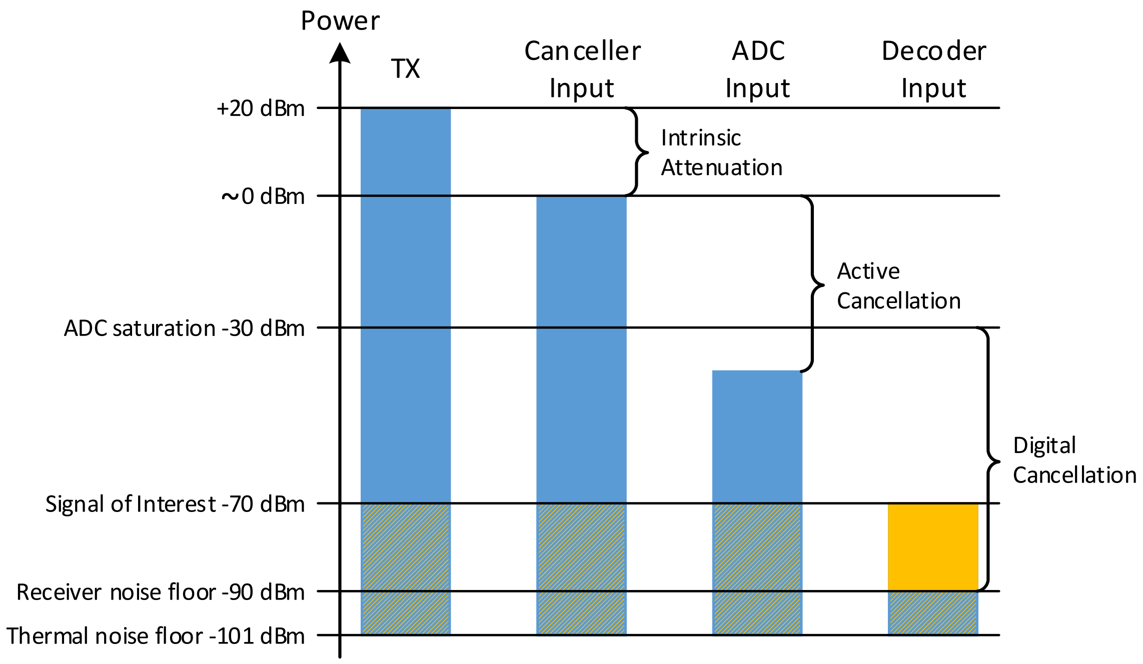


Figure 2.3 *WLAN power levels after each stage of cancellation. The blue represents the transmission signal and the yellow represents the signal of interest.*

sufficiently reduced in the analog domain, the dynamic range of the ADC cannot cover both high-power SI and low-power SoI. The gain of the VGA would have to be reduced in order to prevent ADC saturation. Saturation causes the signal to become clipped, leading to information loss. It follows that the SNR of the SoI is reduced and the spectral efficiency decreases. In the worst case, full-duplex could be less efficient than the traditional half-duplex methods. Simply increasing the isolation between transmitter and receiver is often not a viable option, especially in mobile devices, where the *printed circuit board* (PCB) surface area is very limited. For these reasons it is vital to develop systems capable of actively canceling the SI in the analog domain.

2.2 Self-interference sources

This thesis focuses on circulator based full-duplex operation, where the transmitter and the receiver share the same antenna. Circulator is a device, which allows a signal to pass from one port to the next one, while isolating it from the previous port. In terms of full-duplex this means that the three ports used are TX, antenna and RX. The operation of a circulator and the SI sources are illustrated in Figure 2.4. The TX signal is allowed to pass to the antenna and it is isolated from the RX port. Correspondingly the antenna is isolated from the TX port while passing the signal to the RX port. In reality, however, the isolation

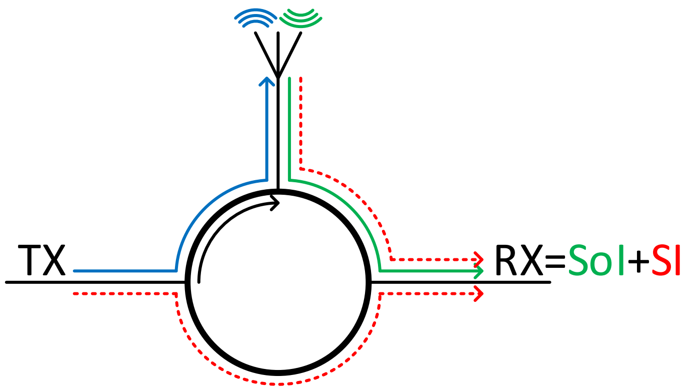


***Figure* 2.4** *Circulator.*

in the opposite direction is not ideal. As a result, the TX signal will leak to the RX port. This is the most direct source of SI. Another source of SI is the antenna reflection. Imperfect impedance matching between the antenna and the circulator will cause part of the signal to be reflected back towards the circulator from the antenna. The source of the remaining SI is reflections from surrounding objects, where the transmitted signal is reflected back to the antenna. While there are countless ways a signal can be reflected back to the antenna, the power of the reflections is attenuated through path loss. Only the closest reflections have a significant effect on the SI power. Essentially this means that the SI is the sum of copies of the TX signal with varying delays and attenuation. Moving the antenna or any object nearby will cause the antenna reflection and reflections from surrounding objects to change. To track these changes, the canceller requires adaptive control.

2.3 RF canceller architecture

The SI is removed by creating a cancellation signal, which is subtracted from the RX signal. It is beneficial to use the PA output as the reference for the cancellation signal, because it allows the canceller to also target the distortion and noise added by the PA [10]. The block diagram of the canceller is presented in Figure 2.5.

The RF signal from the PA output is divided into 4 parts using a 4-way splitter BP4U [11]. One part will be used for the actual TX signal to the circulator, while the remaining 3 parts will be used for the 3 taps of the canceller. A tap is a signal processing path where the canceller modifies the reference signal to match the response of the SI channel as accurately as possible. Each tap consists of a delay line, tap coupler, vector modulator and a downconverter.

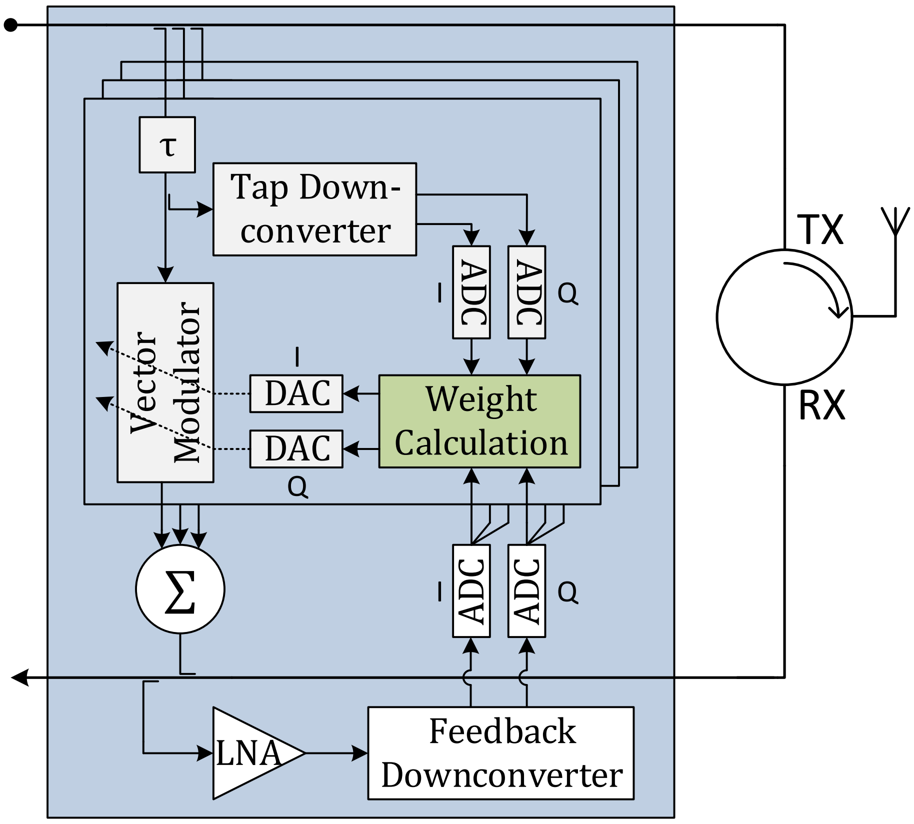


Figure 2.5 Canceller functional diagram. The weight calculation block is discussed in chapter 3.4.

The tap signals are delayed by different amounts to target different SI sources. The delay is created by increasing the distance the signal has to travel. The signals have some inherent delay from the components and transmission lines on the canceller PCB. The tap targeting the static leakage of the circulator has a fixed delay. The delays of the remaining two taps can be controlled by changing the length of a coaxial cable between the input splitter and a tap coupler.

Tap coupler is a 10 dB directional coupler, which divides the tap signal power between a downconverter and a vector modulator. The direct output of the coupler will feed the vector modulator, while the coupled output will go to the downconverter.

Downconversion of the RF signals to baseband is required by the control system. Directly sampling the RF signal is complicated, unnecessary and it would place high requirements for the data acquisition. MAX2023 [12] is a demodulating downconverter, which splits the incoming RF signal into *balanced* baseband IQ signals. The baseband signals are filtered using a LPF to remove any unwanted frequencies above the I and Q bandwidth.

After filtering the Balanced signals are converted to *single-ended* using a *balun*. Notice that the cancellation takes places entirely in RF and the downconversion is only related to the control system. The insertion loss of the downconverter is roughly 11 dB.

Vector modulator is a component, which allows the phase and gain of RF signals to be adjusted [13]. The control system is based on the manipulation of the vector modulators, whose block diagram is shown in Figure 2.6. The gain and phase control are presented in Figure 2.7. The gain and phase are adjusted by two control voltages I and Q. The vector modulator splits the incoming RF signal into IQ components, whose gain is dependent on the specific control voltage. Separately adjusting the amplitude of the signals will result in both phase and gain change in the recombined RF signal.

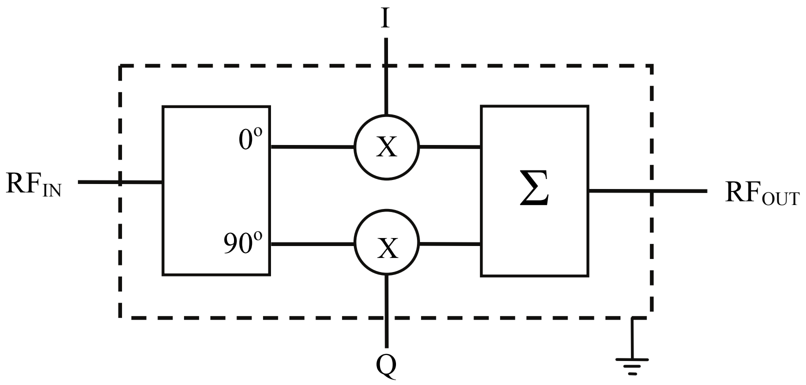


Figure 2.6 *Vector modulator block diagram [13].*

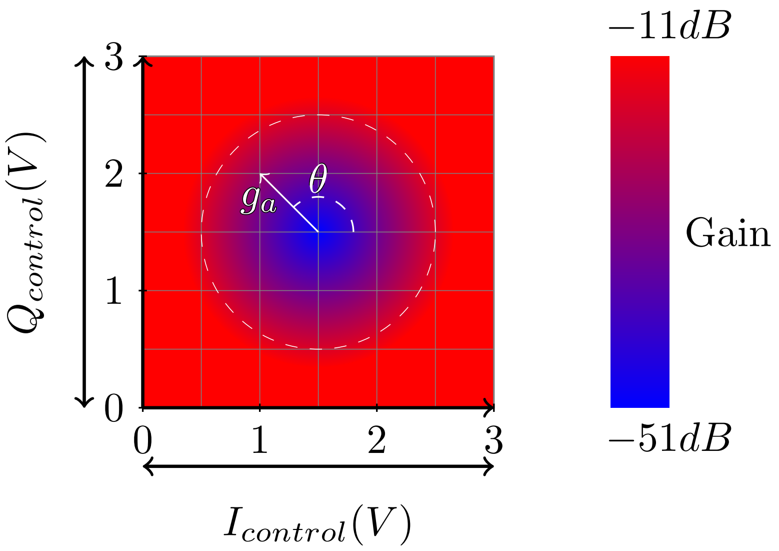


Figure 2.7 *Vector modulator gain and phase control. The white dashed line shows the maximum gain circle after which the change in gain becomes insignificant [14].*

The gain of the vector modulator can be calculated using the following equation:

$$|G| = G_{MAX} \times 2 \times \sqrt{\left(\frac{I_{control} - V_{mi}}{V_{RANGE}}\right)^2 + \left(\frac{Q_{control} - V_{mq}}{V_{RANGE}}\right)^2}, \tag{2.2}$$

where V_{mi} and V_{mq} are the I and Q voltages of the null point respectively and V_{RANGE} is the diameter of the maximum gain circle. The phase change can be calculated using the following equation:

$$\theta = arctan\left(\frac{Q_{control} - V_{mq}}{I_{control} - V_{mi}}\right). \tag{2.3}$$

In an ideal case the vector modulator null point is located at 1.5 V for both I and Q voltages. In reality, however, the location of the null point may vary slightly, but the negative feedback loop of the control system will compensate for the difference. The null point is the origin for the gain vector. It is worth noting that near the null point the phase control is more sensitive to any noise on the control voltages than with higher gain. This is due to the fact that the phase angle is determined by the distance of I and Q controls from the null point. While the noise stays the same, its relative effect becomes larger. For the best phase accuracy the system should be designed so that the vector modulators operate near the maximum gain circle.

The outputs of the three vector modulators are combined using 3-way combiner SCN-3-28 [15] to form the cancellation signal. The cancellation signal is complementary to the SI in the RX. The RX and the cancellation signal are then coupled together, leaving behind the SoI.

The feedback chain of the canceller works similarly to a tap. The canceled signal is coupled using a 10 dB coupler and the coupled output feeds a downconverter. Unlike the taps, there is no need for a vector modulator in the feedback chain and the direct output of the coupler is the output of the canceller.

The canceller layout is presented in Figure 2.9 and the finished board with explanations in Figure 2.8. The layout was designed for 4-layer PCB. The RF components and traces are located on the top layer. First middle layer is reserved for ground only to shield the RF traces. Second middle layer contains the downconverter power supply routing, baseband I and Q signal routing, and vector modulator control voltage routing. The bottom layer contains the vector modulator power supply routing.

Both top and bottom layer have a low-pass filter for each I and Q signal. The filter located on the top layer has a higher cut-off frequency of 40 MHz compared to the 10 MHz of the one located on the bottom layer. Jumpers can be used to pass the baseband signals through either one of the filters depending on the bandwidth of the input signal. The Baluns for the tap signals are also located on the bottom side to prevent them from having to cross the RF traces on the top layer. The Baluns for the feedback chain are located on the top layer. The baseband signals can be taken from the *sub-miniature type-a* (SMA) connectors next to the the feedback chain as Single-ended (after the balun) or from the left and right *RJ45* connectors on the top edge of the board as Balanced (before the balun).

The middle RJ45 connector is used for the vector modulator control voltage input. Optionally the control voltages can be fed to the 2x2 male headers located next to the vector modulators. The 2x2 headers also allow the control voltages to be easily probed using an oscilloscope when RJ45 input is used. The canceller PCB dimensions are $10\,\mathrm{cm} \times 14\,\mathrm{cm}$. The unedited PCB figure is located in the appendix A.

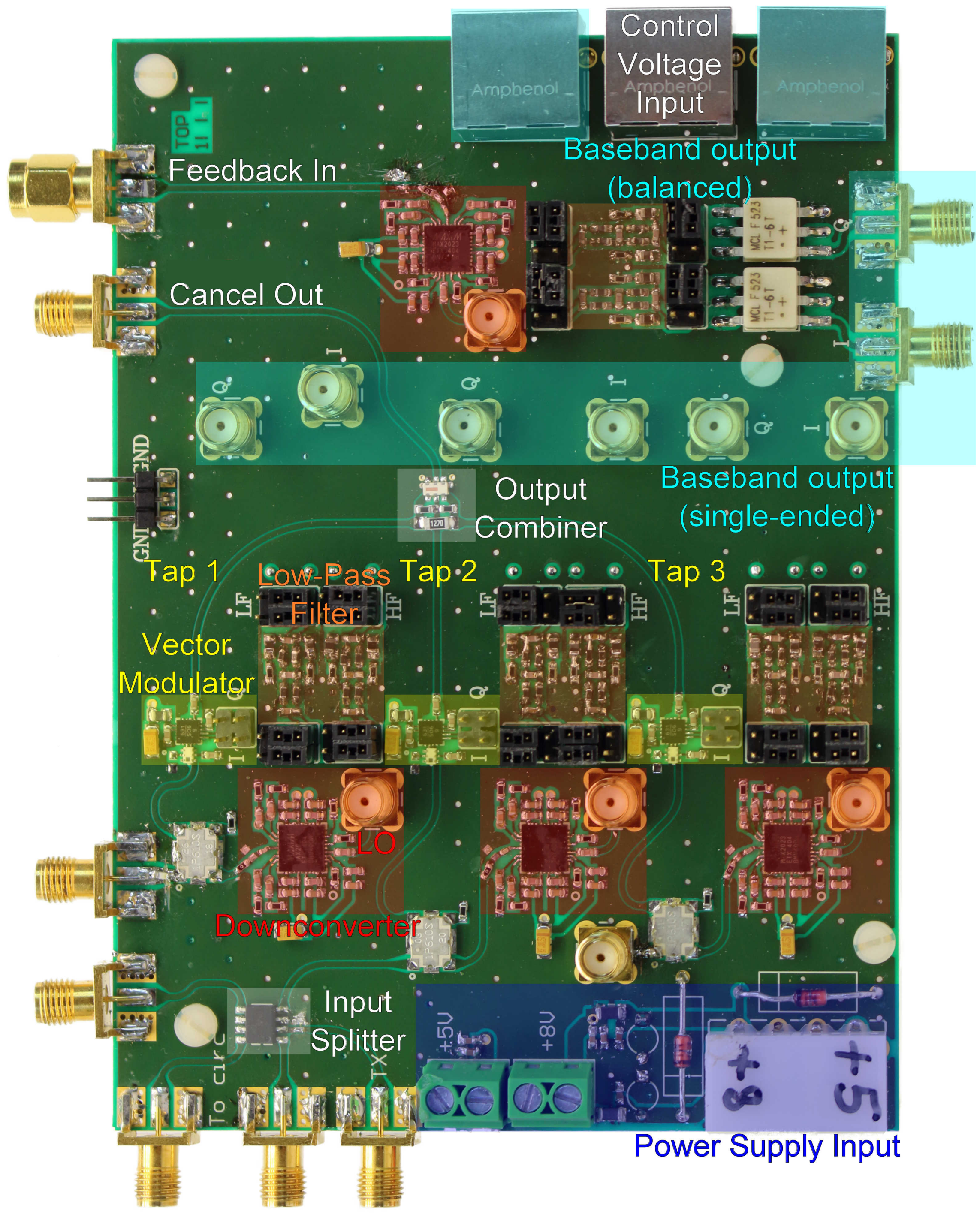


Figure 2.8 *The constructed canceller PCB.*

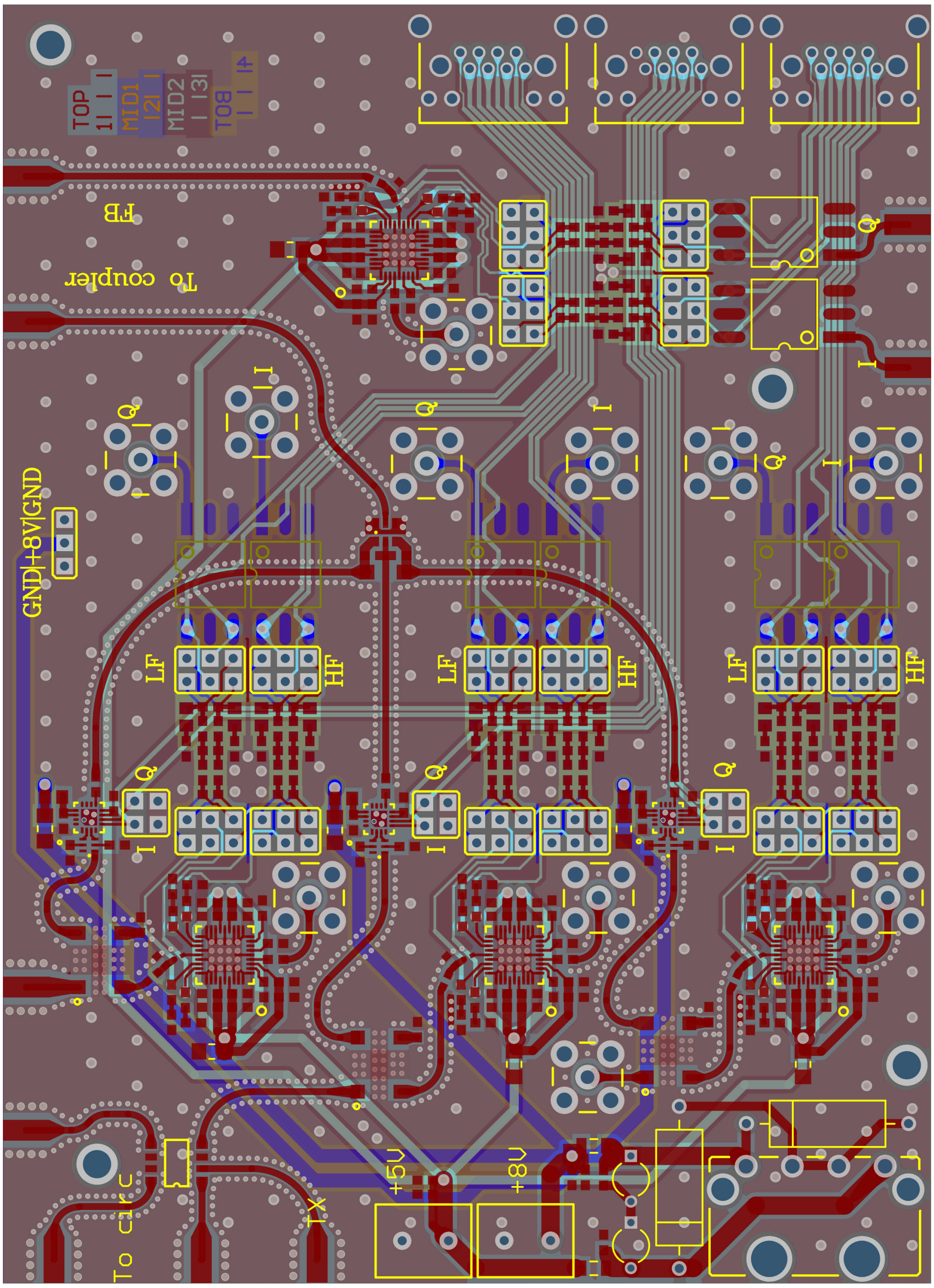


Figure 2.9 *Designed canceller PCB layout.*

2.4 Control algorithm

The control voltages for the vector modulators are calculated using a type of adaptive filter algorithm called *least mean squares* (LMS). The LMS algorithm tries to minimize a cost function by calculating the gradient of the cost function and adjusting the filter weights, values that control the canceller, towards the negative of the gradient. Using the canceled signal as the error signal for the algorithm causes the power of the SI to be minimized. The TX signal is used as an input for the algorithm to calculate the gradient. The SoI is not affected by the filter, because it is not correlated with the algorithm input. The amount by which the filter weights change depends on the power of the error signal and algorithm step-size.

The weights w of the adaptive filter are updated using the following equation:

$$w_n(k+1) = w_n(k) + \mu x_n^*(k)e(k), \tag{2.4}$$

where μ is the step-size, $x_n(k)$ and $e(k)$ are the complex baseband signals of the n^{th} tap signal and the error signal respectively, and k is the discrete-time index. Complex conjugation is denoted by $()^*$. The filter weight updates can be expressed using the baseband IQ signals $x_{n,I}$ and $x_{n,Q}$

$$w_{n,I}(k+1) = w_{n,I}(k) + \mu\Big(x_{n,I}(k)e_I(k) + x_{n,Q}(k)e_Q(k)\Big) \tag{2.5}$$

$$w_{n,Q}(k+1) = w_{n,I}(k) + \mu\Big(x_{n,I}(k)e_Q(k) - x_{n,Q}(k)e_I(k)\Big) \tag{2.6}$$

for the I and Q control respectively. These values must then be converted from digital values to analog voltages, which control the vector modulators. The implementation of the equations 2.5 and 2.6 is illustrated as a block diagram in Figure 3.11.

Essentially the algorithm tries to mimic the response of the SI channels and create corresponding replicas by adjusting the phase and attenuation of the delayed tap signals. Similar methods of adaptive filtering are commonly used, for example, in audio systems, where the output of a loudspeaker is recorded by a microphone. Acoustic echo cancellation is typically done entirely in digital domain instead of analog cancellation hardware.

2.5 Control method and digitization

In prototyping digital control provides multiple benefits compared to analog implementations. Most significant advantage is the flexibility provided by the ability to change the control system through software. With analog circuits, tuning components can be adjusted easily, but any major modification can be impossible to carry out without redesigning the entire circuit. With digital systems, modifications or even completely different algorithms can be implemented using the existing hardware. For prototyping purposes this is highly beneficial as it reduces the development time. It is also easier to transfer the design from software simulations into a digital processing system, than it is to create an analog circuit from the ground up. This also allows the software to be developed in parallel with the hardware.

In order to process signals digitally they must first be converted from analog signals to digital values. As a result, the continuous analog signal becomes quantized. The parameters of the ADC dictate how accurately the signal can be converted. The difference between the largest and smallest quantifiable input signals is known as dynamic range. Dynamic range is usually expressed in decibels. Closely related to the dynamic range is the resolution of the ADC, which defines the smallest change in the analog input that can be measured over a specific voltage range with finite number of digital values. *Least significant bit* (LSB) represents the difference between two consecutive digital values. The voltage represented by the LSB can be calculated from the converters full-scale voltage range V_{FS} and number of bits N with the following equation:

$$V_{LSB} = \frac{V_{FS}}{2^N - 1}. \tag{2.7}$$

For an ideal 12-bit converter with 2 V full-scale voltage range, the resolution is 488.4 µV. The LSB also defines the quantization error caused by the rounding the analog value to the nearest corresponding digital value. The range of the quantization error is $\pm\frac{1}{2} \times LSB$, when rounding to the nearest number. If the result is rounded down the quantization error will range from 0 to $-LSB$. The difference in quantization error between 3- and 4-bit digitization is depicted in Figure 2.10.

SNR describes the the ratio between the signal power and the power of any noise present in the system

$$SNR = \frac{P_{signal} + P_{noise}}{P_{noise}}. \tag{2.8}$$

For an ideal ADC the SNR is limited by the quantization noise. Theoretical value for

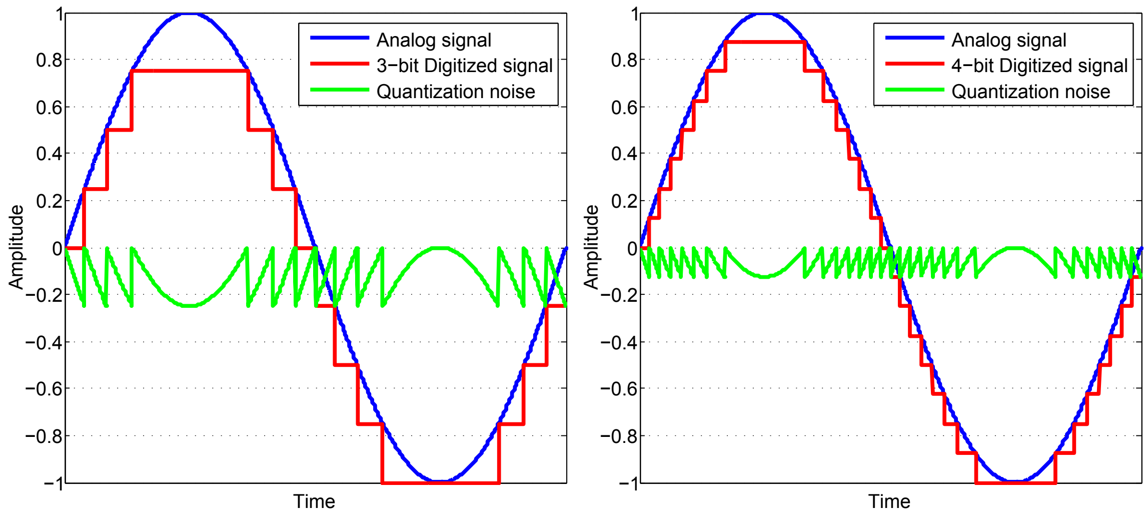


Figure 2.10 *A comparison between 3- and 4-bit digitization of a full-scale sine wave. The digitized value is rounded down.*

SNR can be calculated from the number of bits N

$$SNR = 6.02 \times N + 1.76. \tag{2.9}$$

In practice, however, there is additional thermal noise, clock jitter and reference noise, which reduce the SNR further.

A better measure of an ADC's performance is the *signal-to-noise-and-distortion ratio* (SINAD), which takes into account the power of the distortion components

$$SINAD = \frac{P_{signal} + P_{noise} + P_{distortion}}{P_{noise} + P_{distortion}}. \tag{2.10}$$

The *effective number of bits* (ENOB) can be calculated from the SINAD

$$ENOB = \frac{SINAD - 1.76}{6.02}. \tag{2.11}$$

The SNR can be improved by oversampling the input signal and decimating it. When the signal is oversampled, the power of the quantization noise is spread over a wider bandwidth, and can then be low-pass filtered. The improvement of the SNR can be calculated from the ratio of sampling frequency and Nyquist frequency:

$$SNR_{gain} = 10 \times log_{10}(\frac{Fs}{F_{Nyquist}}), \tag{2.12}$$

where $F_{Nyquist}$ denotes the Nyquist frequency and Fs a power of 2 multiple of $F_{Nyquist}$. This means that doubling the sampling frequency can improve the SNR by 3 dB, which means that the reduction in quantization noise is equal to having extra 0.5 bits in the conversion. Oversampling by a factor of 4 would improve the SNR by 6.02 dB and ENOB by 1. Decimation also reduces the data rate, which is useful when the signal processor input data rate is limited.

2.6 FPGAs in digital signal processing

In a static environment the optimal filter weights for the canceller remain fairly constant over time. This allows higher latency systems such as computers to be used along with more complex algorithms that are computationally expensive. However, for a typical indoor wireless access point the SI channels would be varying almost constantly. The variation of the SI channels is even more prominent in mobile devices, where the user moves the antenna itself. Additionally, in handheld devices, the positioning of users hand can greatly affect the characteristics of the antenna. In order to maintain sufficient cancellation performance in a dynamic environment, the filter weights must be constantly updated. The update also has to happen simultaneously to the cancellation.

FPGAs are a natural choice for a high data-rate, low-latency, multichannel digital signal processing system. FPGAs are integrated circuits with configurable interconnects and logic blocks. The implementation of logic blocks depends on the manufacturer, but a typical logic block is comprised of look-up tables, carry logic and registers. Field-programmable refers to circuits ability to be reconfigured. Through programming of the look-up tables and interconnects, complex digital systems can be synthesized. Most FPGAs also include hard digital signal processing blocks, such as multipliers and floating-point operators. Synthesizing common operations needed in digital signal processing using normal logic blocks can often take a lot of resources and reduce the maximum performance of the system.

In real-time signal processing a data flow is continuously processed at a rate higher than the sample rate of the input. FPGAs excel at handling data flows due to their intrinsic pipelining. Pipelining describes FPGAs' ability to handle a constant stream of inputs at a high rate by splitting the process into multiple small stages using registers. Data travels through combinatorial logic from one register to the next one on every clock cycle. A single processing path can contain any number of registers between input and output and still provide a throughput of one sample per clock cycle. Another implication of the real-time signal processing on FPGAs is deterministic latency, which can be calculated as the product of clock rate and number of registers between input and output. For these reasons FPGAs make an excellent choice for the platform to implement the control system of the canceller on.

FPGAs operate at lower clock frequencies than traditional microprocessors. However, FPGAs allow the designer to create multiple parallel channels, one for each input, increasing the throughput significantly. As a result, FPGAs do not suffer any performance penalty from increased input channel count and they are only limited by the amount of available *input/output* (I/O) and logic resources. On a microprocessor, the processing power is limited. Increasing the number of channels will decrease the throughput per channel due to the fixed hardware architecture.

The negative aspect of FPGAs is the difficulty of general purpose processing. Processing data like text strings or complex data structures is cumbersome to implement at a low level. It is common for manufacturers to add a hard microprocessor to the integrated circuit along with the FPGA to handle the general purpose processing and use the parallelism of the FPGA for specific functions. Another option is to synthesize an entire microprocessor using the FPGA logic resources. These are called soft microprocessors. Soft microprocessors typically run on a slower clock, but their features can be customized to match the specific needs of the design. FPGA development is slower compared to microprocessors, which can utilize high-level programming languages. In return FPGAs offer great versatility and optimization possibilities.

3. CONTROL SYSTEM IMPLEMENTATION

In this chapter the hardware and software implementations of the control system are discussed. The control system utilizes the baseband signals from the canceller to create control voltages for the vector modulator. An adaptive filtering algorithm is used in the digital signal processing. The system includes an ADC, a DAC and an FPGA. A user interface is also explained briefly. The block diagram of the control system is presented in Figure 3.1, where the scope of this chapter is marked with a dashed red line. A more detailed block diagram of the implementation inside the FPGA is depicted in Figure 3.2.

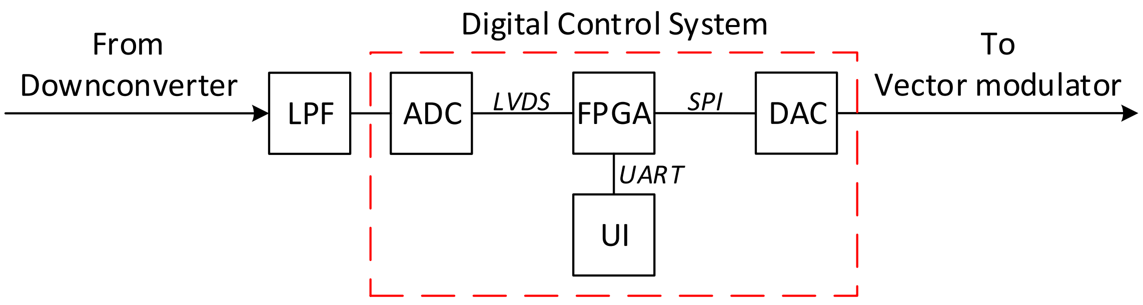


Figure 3.1 *High-level block diagram of the control system. The analog anti-aliasing LPF is located on the canceller board as mentioned in previous chapter.*

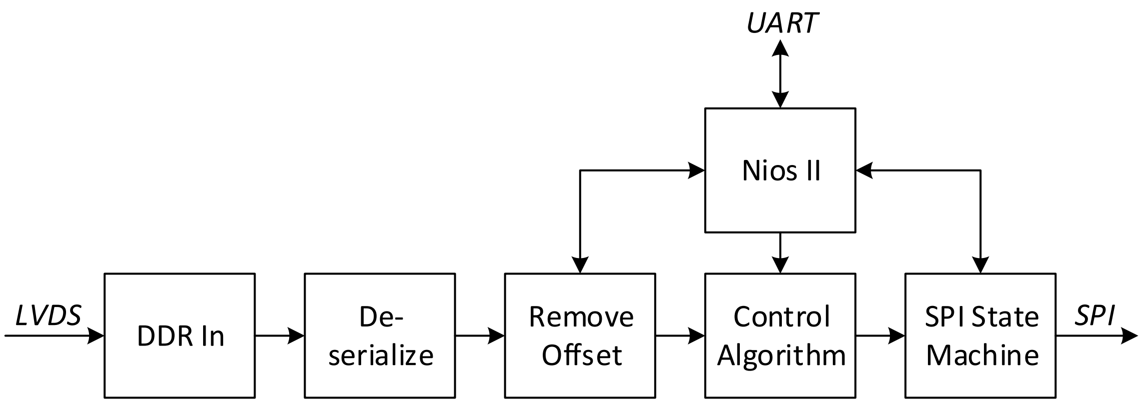


Figure 3.2 *Block diagram of the FPGA implementation.*

A BeMicroCV-A9 [16] FPGA development board from Arrow was chosen for this project. It contains Cyclone V E FPGA (5CEFA9F23C8). The FPGA is programmed using *very* *high speed integrated circuit hardware description language* (VHDL). Both VHDL implementation and hardware are also fully compatible with the older version BeMicroCV-A2 [17], which has a smaller version of the Cyclone V E FPGA. Figure 3.3 shows the newer version of the development board.

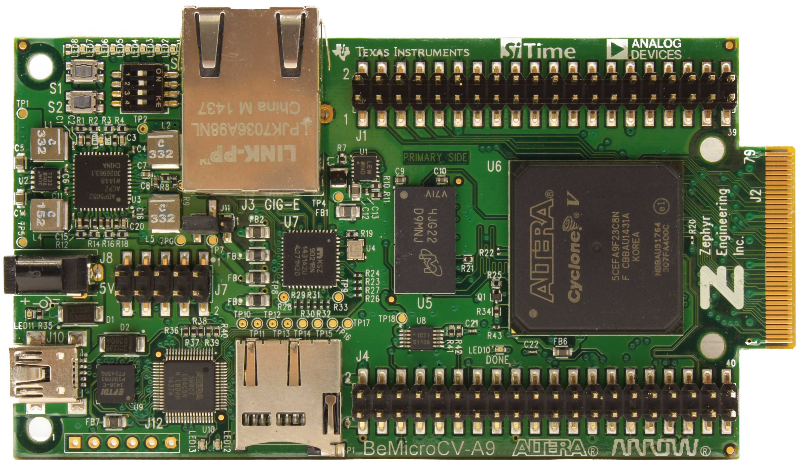


***Figure* 3.3** *BeMicroCV-A9.*

The VHDL development was done on *Quartus II* and the VHDL code was simulated and verified using *ModelSim*. PCB schematics and layouts were designed using *Altium Designer*.

3.1 Analog-to-digital conversion

The analog-to-digital conversion of the baseband signals from the downconverters is done using a Texas Instruments ADS5295EVM [18] evaluation module, which is displayed in Figure 3.4. Each of the downconverters on the canceller requires two ADC channels for the IQ signals. There are 4 downconverters, 3 for the taps and 1 for the feedback resulting in a minimum requirement of 8 ADC channels.

The evaluation module utilizes 8-channel ADS5295 ADC [19]. This ADC was chosen because it provides high sampling rate of up to 80 MSPS with one-wire interface. In this case one-wire refers to the number of signal pairs per digital output channel, not the physical number of traces. Two-wire interface would enable sampling rate of 100 MSPS and allow faster data transfer as the data is split over two signal pairs, but it would also require extra receiver channels and PCB area. The conversion mode can be chosen between 10 and 12 bits. The ADC has an internal voltage reference of 2 V with an accuracy of ±30 mV.

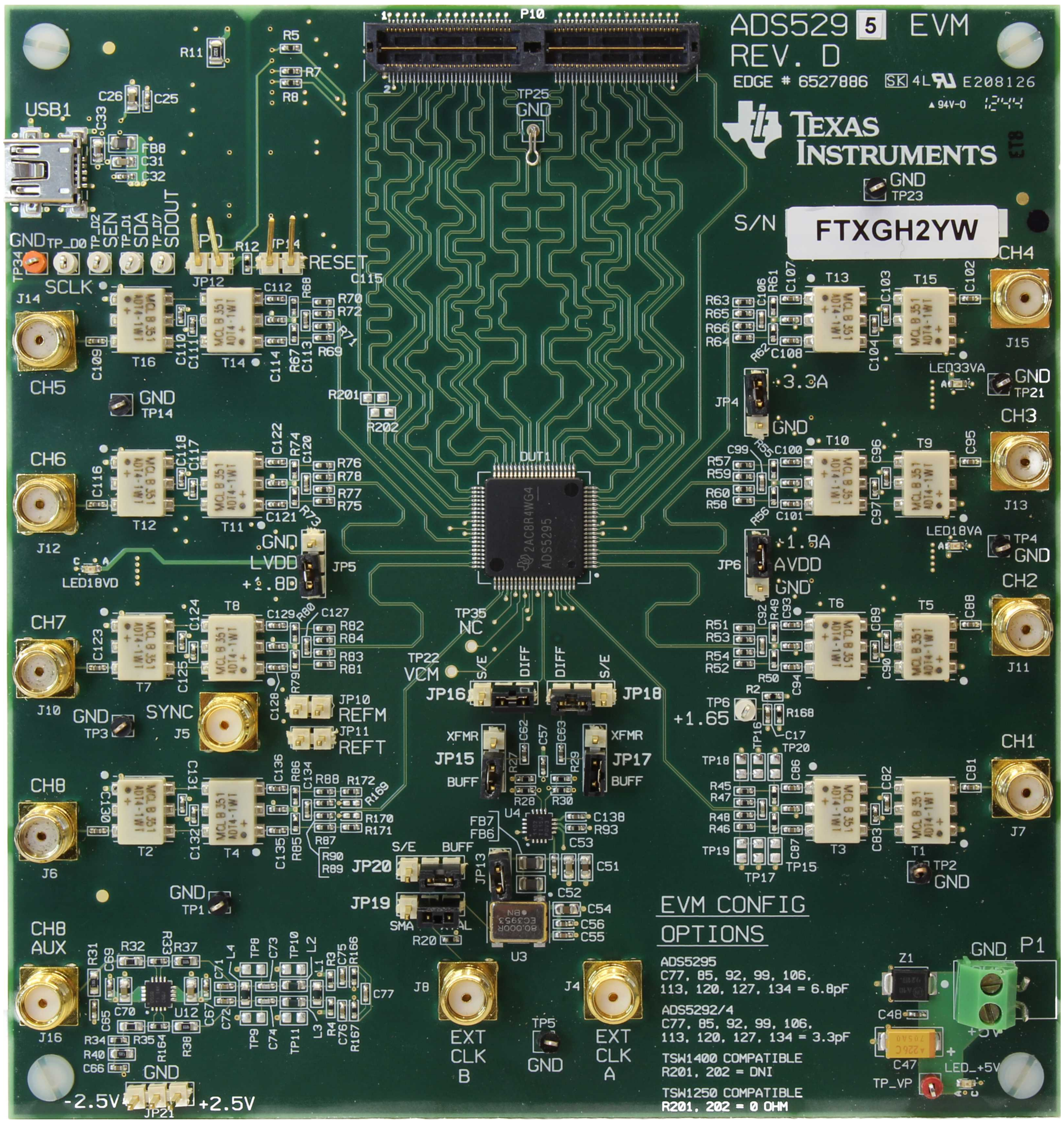


Figure 3.4 *ADS5295EVM.*

The ADS5295 transmits the digitized samples using a high-speed *low-voltage differential signaling* (LVDS) interface. Each data line on the LVDS interface consists of a signal pair that carries the data as voltage difference between the traces. An LVDS transmitter drives a constant current of 3.5 mA through the receiver's 100 Ω termination, resulting in voltage of 350 mV over the termination. The direction of the current dictates the polarity of the differential signal. The Cyclone V FPGA of the BemicroCV-A9 provides multiple *on-chip termination* (OCT) options, including 100 Ω differential input termination required by the LVDS. Compared to external termination, OCT saves PCB area and allows the termination to be closer to the receiver.

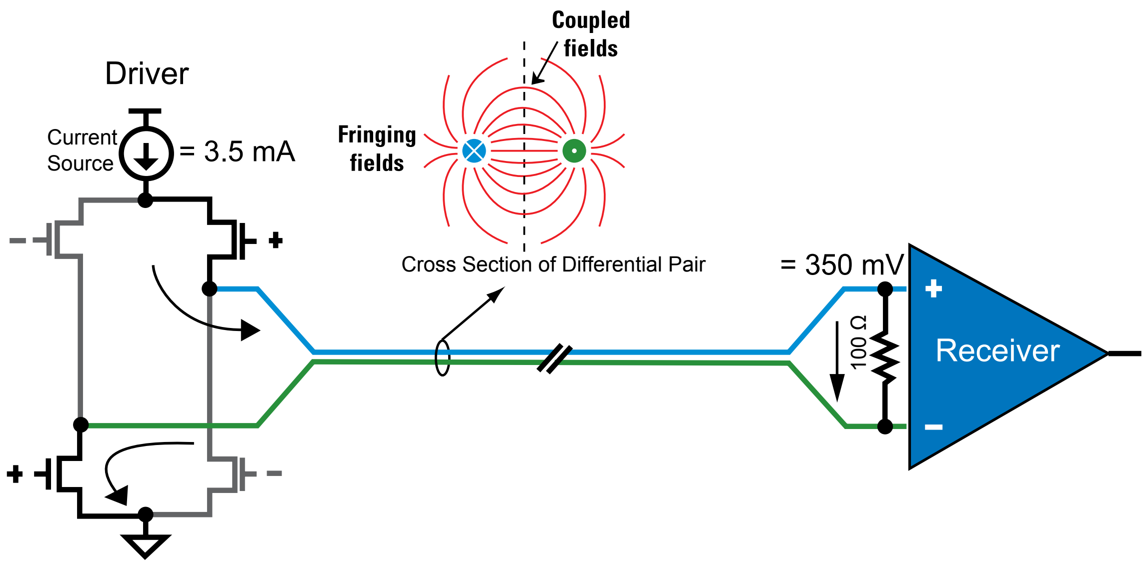


Figure 3.5 *LVDS Driver and receiver [20].*

One of the limiting factors in the precision of the digital control chain is the dynamic range and reference level of the ADC. 12-Bit conversion mode was chosen to maximize the dynamic range. The power of the feedback signal is attenuated by 10 dB in the feedback coupler and another 11 dB in the feedback downconverter. The dynamic range of the ADC is 70.6 dBFS, where the reference level is 10 dBm (2 $V_{p\text{-}p}$ in 50 Ω system). This means that without any amplification, the feedback signal is lost under the ADC's noise floor after RX signal reaches -39.6 dBm. An LNA with a gain of 22 dB was added to the feedback chain before the downconverter, so that higher amount of cancellation could be reached before the feedback signal power becomes too low. Another option would be to use *automatic gain control* (AGC) in the feedback chain to maximize the use of the available dynamic range. However, using AGC would require adjusting the step-size of the algorithm, inversely proportional to the amount of gain, to prevent instability from overshoot and oscillation. With constant gain, the step-size can be kept constant because lower power in the feedback intrinsically reduces the size of the steps by the algorithm.

The ADS5295EVM provides an 80 MHz on-board clock [18]. Typically the frame clock is equal to the sampling rate of the converter, which would mean $80\,\mathrm{MSPS} \times 12bits = 960\,\mathrm{Mbps}$ data rate per channel. However, the Cyclone V E with a speed grade of C8, such as the one on the BeMicroCV-A9, is capable of receiving a maximum of 640 Mbps per channel using LVDS, which means a maximum frame clock of 53.3 MHz with 12-bit frames [21, p. 46]. The ADS5295 offers programmable on-board digital processing blocks such as *finite impulse response* (FIR) decimation filter and *infinite impulse response* (IIR) high-pass filter. Decimate-by-2 filter was used to lower the output sample rate from 80 MSPS to 40 MSPS, which is within the FPGA receiver limits. Using higher sampling frequency and then low-pass filtering and decimating also improves the ADC dynamic performance as explained in chapter 2.5. The change in dynamic performance and data rate is presented in Table 3.1. The values are taken from the datasheet of the ADS5295 [19, p. 15;20].

***Table 3.1** ADS5295 dynamic performance before and after decimation.*

Filtering	SNR	SINAD	ENOB	Data rate (80 MHz)
No decimation	70.8 dBFS	70.4 dBFS	11.4 bits	960 Mbps
Decimate-by-2	73.9 dBFS	73.3 dBFS	11.9 bits	480 Mbps

The ADS5295, like most high-speed ADCs, uses a pipeline architecture, where the sample is converted using multiple low resolution stages. The pipeline causes a 12-clock cycle latency. For 80 MSPS sampling rate, the added delay will be 150 ns.

The ADS5295 has a fairly high offset error range, $\pm 20\,\mathrm{mV}$. Removing the offset error from the conversion result is extremely important. It has been shown that the offset present in the input signals can significantly reduce the performance of the LMS algorithm [22]. The actual offset in each channel can be calculated by taking the mean of the incoming signal. Once the offset value is known, it can be subtracted from the incoming values. In the VHDL implementation the offset is calculated from the mean of 2^{16} successive samples for each channel separately.

3.2 Adapter board

An adapter board was built to connect the data converters to the FPGA. The ADS5295 has 18 LVDS output channels. Even though the FPGA on the BeMicroCV-A9 has enough differential inputs, not all of them are routed to the connectors. For this reason, only nine of the ADS5295 LVDS channels are used, which prevents the use of two-wire mode for higher data rate. To minimize intra-pair and pair-to-pair skew, all of the LVDS trace lengths are matched within 0.3 mm of each other.

The constructed adapter board is presented in Figure 3.6. On the top side of the board are two 40-pin female connectors, which attach to the two 40-pin male headers of the

BemicroCV-A9 displayed in Figure 3.3. Located on the bottom side is the connector to the ADS5295EVM, whose mating connector is visible in the top of Figure 3.4. The DAC is located on the top side along with two 2x2 headers for power input and *universal asynchronous receiver transmitter* (UART). There are two optional ways to connect the DAC outputs to the vector modulator control inputs. Twisted pair cables can be used to connect the DAC 6x2 header to the 2x2 headers next to the vector modulators shown in Figure 2.8. However, the preferable option is the RJ45 connector located on the bottom side of the board. The RJ45 allows a shielded category 6 cable to be used, which is more robust against *electromagnetic interference* (EMI). The DAC is discussed in higher detail in chapter 3.5.

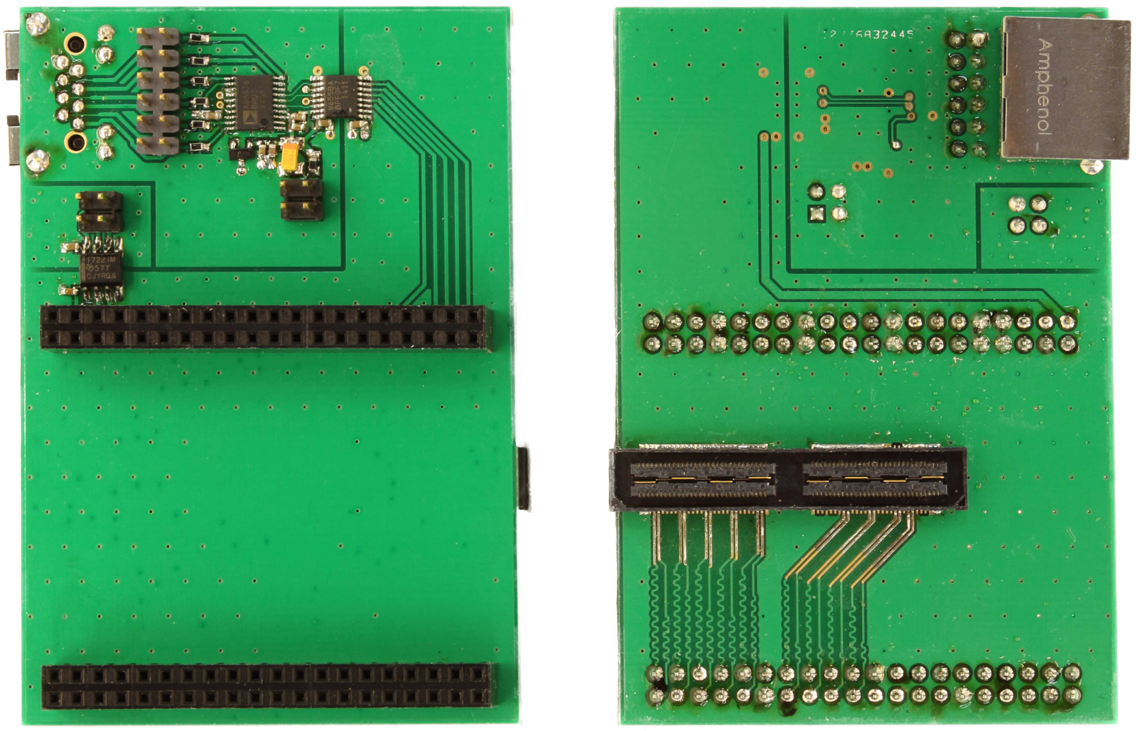


Figure 3.6 *Left: adapter board top side. Right: adapter board bottom side.*

To prevent ground loops and to reduce the noise in the control voltages, the DAC was isolated from the FPGA using a 5/0 SI8655BA-B-IU [23] digital isolator by Silicon Labs. The 5/0 refers to the amount of channels and each direction. In this case there are 5 channels that are all in the same direction, because there are no signals coming back from the DAC. The data rate of the digital isolator is 150 Mbps, which is enough for a 50 MHz *serial peripheral interface* (SPI). The voltage isolation rating is irrelevant as it is only used to isolate the noisy FPGA from the DAC. The SI8655 uses on-off keying to pass the information through an isolation barrier. An RF signal is modulated based on the state of the binary input signal. The RF signal passes through the capacitive isolation barrier. On the other side a demodulator drives the output based on the energy of the RF signal. [24]

The UART was also isolated from the FPGA using a 1/1 (one channel in each direction) Texas Instruments ISO7221MDR [25] digital isolator with a data rate of 150 Mbps. As with the SI8655BA-B-IU the isolation rating is not important because it is only used to isolate the PC, running the user interface, from the measurement setup. Unlike the SISI8655, the ISO7221 uses edge-based communication. A single ended binary signal is split into a differential signal pair using an inverter. The transients of the differential signal behind the isolation barrier are detected using comparators. The outputs of the comparators drive a NOR-gate latch, which drives the isolated signal. [24]

Figure 3.7 displays the ADS5295EVM and BeMicroCV-A9 connected with the adapter board in between. The RJ45 connector for the analog outputs of the DAC can be seen on the left side on the bottom of the adapter board.

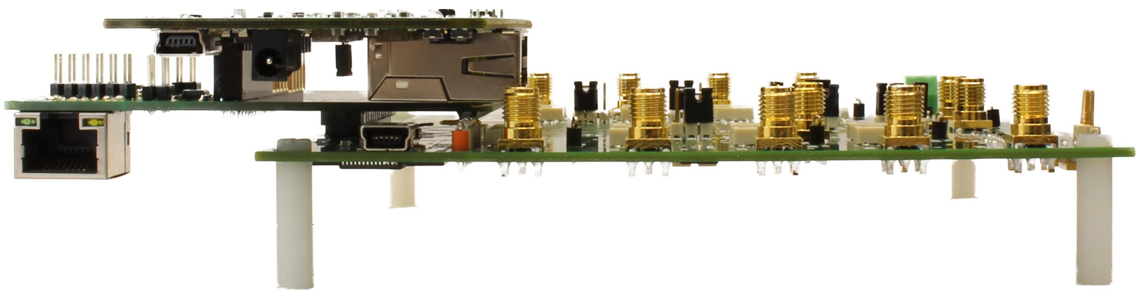


Figure 3.7 *From the bottom to the top: the ADS5295EVM, the adapter board and the BeMicroCV-A9.*

3.3 Data capture and deserialization

The data transfer between the ADC and the FPGA requires two clocks, a frame clock and a bit clock, in addition to the data channels. The frame clock is a slower clock, whose rising edge indicates when a sequence, or a frame, of bits starts. The bit clock indicates when the receiver should sample the signal on the data channel. For *single data rate* (SDR) transfer, the data is sampled either on the rising or the falling edge of the bit clock. With *double data rate* (DDR) transfer, the data is sampled on both rising and falling edges of the bit clock. Therefore, the clock frequency for DDR transfer is halved compared to SDR. Lower frequency clock signal suffers less signal degradation.

The Cyclone V LVDS receiver clock must be routed from the same I/O-region as the data lines [26, c. 5, p. 12]. The 9 LVDS pairs on the BeMicro CV are split between two I/O regions, therefore, the LVDS IP core could not be used and deserialization had to be manually implemented. The LVDS receivers on the FPGA support a maximum deserialization factor of only 10, while 12 was needed to utilize the full range of the ADC. To overcome these limitations an LVDS receiver was implemented using DDR receivers to capture the high speed data and a shift register to deserialize the captured data.

The DDR receiver samples the first bit on falling edge and second bit on the rising edge of the bit clock. The timing of the DDR receiver is shown in Figure 3.10 and the block diagram in Figure 3.8. The output of the DDR receiver, which contains both received bits, is updated on the rising edge of the bit clock.

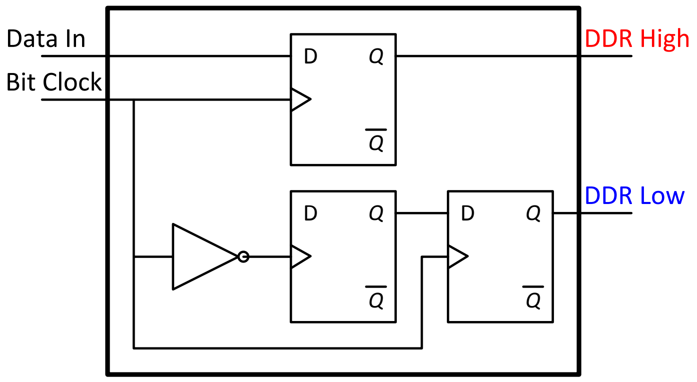


Figure 3.8 *DDR receiver block.*

Because the data is transferred as a sequence of bits, deserialization is needed. Shift register is a structure, which allows serial data to be converted to parallel frames. The frames of the ADC consist of 12 bits, so two shift registers with the depth of six are needed. First register shifts the bits sampled on the falling edge of the bit clock and second register shifts the bits sampled on the rising edge. The behavior of the DDR shift register is illustrated in Figure 3.9. Once all bits are in the shift register, they can be read simultaneously to create the parallel data frame.

Both the frame clock and the bit clock are transmitted through the LVDS along with the data. An eight channel transmitter normally uses 10 signal pairs, however, the Bemicro CV 40-pin header only had 9 LVDS signal pairs available. Naturally none of the data channels could be omitted, but the frame clock already contains the information of the bit clock as long as the data rate is known. For 12-bit conversion with a sample rate of 40 MSPS the bit clock is 480 MHz for SDR transfer and 240 MHz for DDR transfer. The bit clock can be generated on the FPGA with a *phase-locked loop* (PLL), but the phase in relation to the frame clock has to be adjusted. Also the FPGA system clock is generated with the PLL. Figure 3.10 shows how the generated clocks have to be aligned.

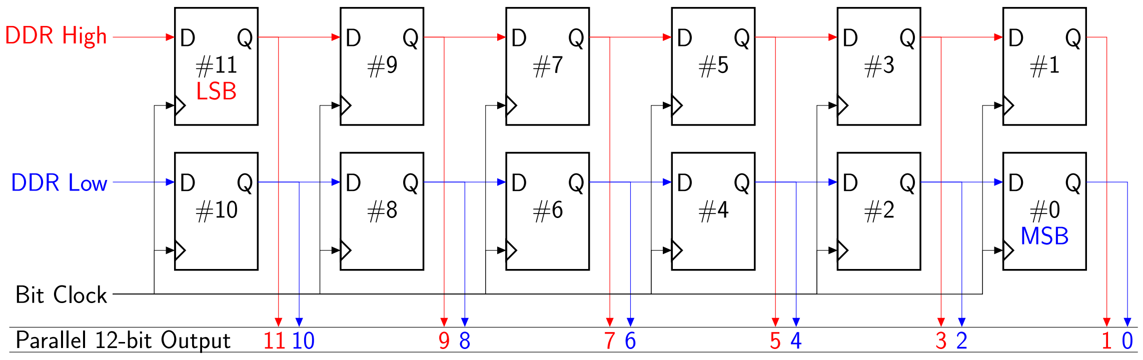


Figure 3.9 *Register-transfer level representation of the implemented shift register.*

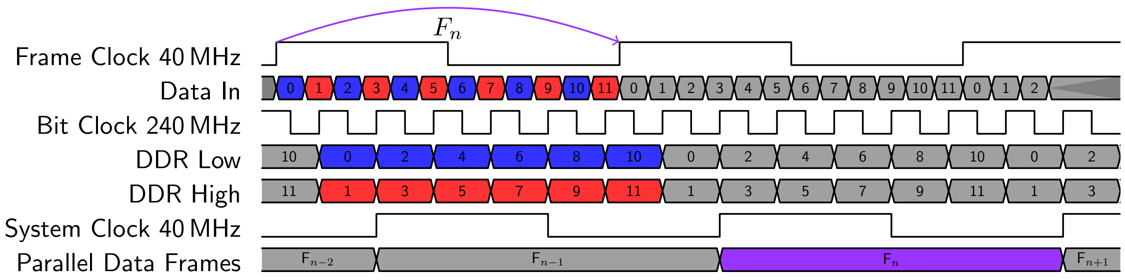


Figure 3.10 *DDR transfer and deserialization timing diagram.*

The PLL is configured based on a reference input clock, which in this case is the ADC frame clock. If we consider the time of the rising edge of the frame clock to be t_0, the rising edge of the generated bit clock happens at time $t_0 - \frac{1}{4\times f_b}$, where f_b is the frequency of the bit clock. The DDR receiver samples the first bit of the frame at $t_0 + \frac{1}{4\times f_b}$, on the falling edge of the bit clock. The second bit is sampled and output is updated at $t_0 + \frac{3}{4\times f_b}$, on the rising edge of the bit clock. The PLL mode is set to LVDS, which compensates for any delay difference between the clock and data pin paths before the DDR register. The PLL lock signal determines whether the reference clock and feedback clock of the PLL are within the lock circuit tolerance. The PLL lock signal is used as a system reset signal because unlocked PLL would cause unpredictable behavior. [27]

The system clock that runs the algorithm and the DAC must be of the same frequency as the frame clock, but its rising edge must be synchronized to the first rising edge of the PLL generated bit clock after a frame has been received. This allows the parallel output of the shift register to be sampled at the correct time. If the rising edges of the bit clock and FPGA clock are misaligned, the signal might not have enough time to propagate from the output of a register into the input of the next one without violating the setup time requirements and possibly corrupting the data.

The VHDL processes implementing the deserialization are presented in Program 3.1. The input bit vectors, `data_in_l` and `data_in_h`, correspond to the `DATA Low` and `DATA High` signals presented in figures 3.10 and 3.9. The clocks `sclk` and `pclk` are the 240 MHz bit clock and the 40 MHz system clock respectively. A 96-bit wide vector `data_out_reg` is needed to hold a 12-bit frame of each channel. On every `sclk` rising edge, the `data_out_reg` is shifted to the left by two and two new bits are added to the LSBs of each channel, which is the function of the shift register. After the last two bits of a frame have been added to the vector, it is clocked to the output `data_out` on the rising edge of `pclk`.

Program 3.1 *VHDL implementation of deserialization.*

```
-- Process to deserialize the signals coming from DDR receiver
PROCESS(sclk,rst_n)
BEGIN
  if rst_n = '0' then
    -- Set output to all zeros after reset
    data_out_reg <= (others => '0');
  elsif rising_edge(sclk) then
    for i in 0 to 93 loop
      -- Shift all bits left by two
      data_out_reg(i+2) <= data_out_reg(i);
    end loop;
    -- For each channel loop
    for i in 0 to 7 loop
      -- Place incoming bits into the LSBs
      data_out_reg(i*12+1) <= data_in_l(i);
      data_out_reg(i*12)   <= data_in_h(i);
    end loop;
  end if;
END PROCESS;

-- Process for the parallel output
PROCESS(pclk, rst_n)
BEGIN
  if rst_n = '0' then
    data_out <= (others => '0');
  elsif rising_edge(pclk) then
    -- Move the parallel frames to the output
    data_out <= data_out_reg;
  end if;
END PROCESS;
```

3.4 Algorithm

The VHDL implementation of the algorithm shown in section 2.4 is fairly straight forward. Each operation in the algorithm was given in their own process in separate files. This allows each operation to be separately tested. Simulating the entire algorithm is difficult due to the lack of any feedback signal. Keeping different operations separated also improves reusability of the code. The required operations are complex conjugation, complex multiplication, bitwise shift and accumulation. The block diagram of the algorithm is presented in Figure 3.11. This implements the mathematical functionality of the equations 2.5 and 2.6.

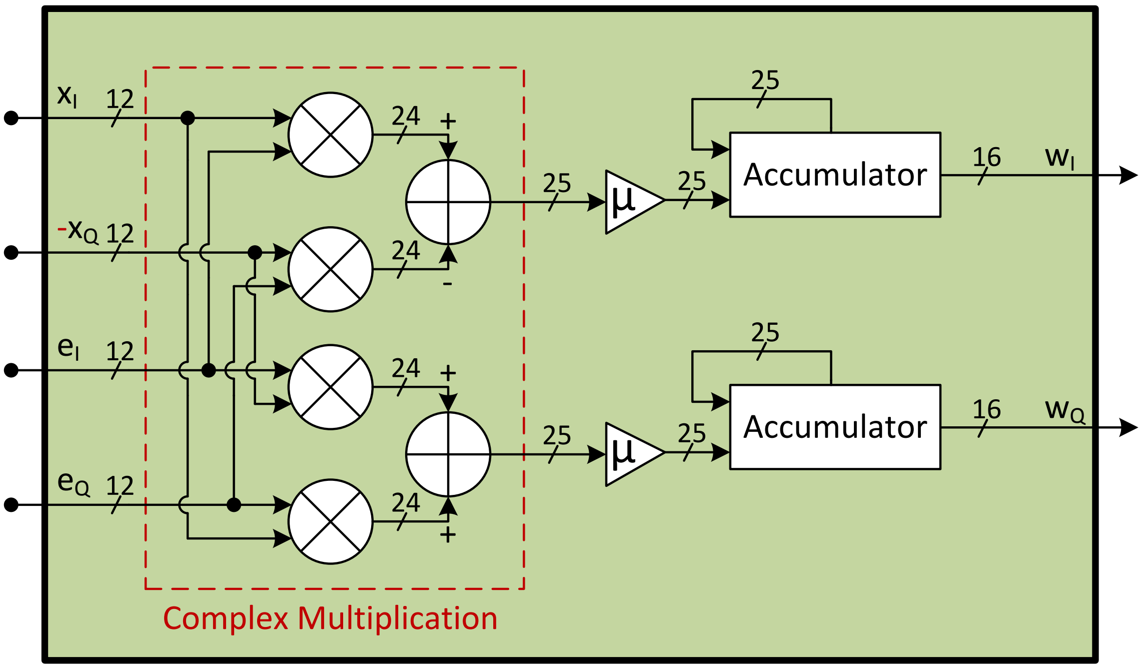


Figure 3.11 *Algorithm block diagram and the signal widths. The $x_{I,Q}$ and the $e_{I,Q}$ are the baseband IQ signals of the tap and feedback signal respectively. The w_I and w_Q are the computed filter weight values for the vector modulator. Note the sign on the x_Q input, which changes as a result of the complex conjugation of the tap signals.*

Complex conjugation is done by changing the sign of the imaginary part of a complex number. The real and imaginary parts of a complex number are represented as two's complement, which means that changing the sign requires inverting of the bits and adding one. Because the range of a two's complement number is from -2^{n-1} to $+2^{n-1} - 1$, where n is the amount of bits, changing the sign on the negative maximum would cause overflow. The overflow would effectively cancel the operation as the result would again become the negative maximum instead of positive maximum. The overflow is prevented by checking if the number is at negative maximum and then just inverting the bits without adding 1, which results in the positive maximum of a two's complement number. The error caused by omitting the addition is $\frac{1}{2^{15}}$, which is negligible. It is also unlikely that any of the complex conjugation operations will reach saturation. Assuming TX power of 20 dBm, the tap signals will be reduced 10 dB in the tap coupler and 11 dB in the downconverter resulting in −1 dBm. Accounting for a PAPR of 10 dB, the peak power of 9 dBm is still lower than the 10 dBm reference voltage. The VHDL code for complex conjugation is presented in Program 3.2.

Program 3.2 *VHDL code for complex conjugation.*

```
result_real  <= data_real;
result_imag  <= (not data_imag) when data_imag = negative_max else
                (not data_imag)+1;
```

For the complex multiplication Altera's megafunction for complex multiplier was used. Megafunctions are parametrized modules that implement a certain function. For complex multiplication the chosen parameters were 12-bit input width, 25-bit output width and output latency of 1 clock cycle. As shown in Figure 3.11, the complex multiplication requires 4 multipliers and 2 adders. Multiplying two 12-bit numbers results in a 24-bit number and a sum of two 24-bit numbers results in a 25-bit number.

The step-size, μ, is controlled with bitwise shifting using negative powers of 2. Each shift to the right is equal to division by 2. This means that the range of the μ is 2^{-n}, where n is an integer between 0 and 16. For 16-bit number, shifting by 16 effectively means the result becoming 0, which stops the control voltages at their current values because the accumulator input becomes 0. Arithmetic shifting is used to maintain the correct sign. The leftmost positions are filled with previous *most significant bit* (MSB): 0 for positive numbers and 1 for negative numbers. Bitwise shifting was chosen over regular division because it provides enough possible values for μ, while consuming much less logic than a normal division operation. The VHDL code for step-size implementation is presented in Program 3.3.

Program 3.3 *VHDL code for step-size control using bit shift.*

```
result <= shift_right(data, to_integer(shift_count));
```

The accumulation is done with 25-bit values, which is the width of the result of the 12-bit complex multiplication. The output of the accumulator is truncated to 16 bits, which is the width of the DAC. Because the accumulator simply sums up incoming values, checking for overflow is extremely important. In the worst case, overflowing accumulation would cause the vector modulator phase control to be shifted 180° and increase the SI instead of canceling. Fortunately, detecting and preventing accumulator overflow is easy using saturation arithmetic. If the sign of the two values to be summed are equal but the sign of the result is the opposite, then overflow would have happened. In this case the result of the accumulator is set to the maximum value with the same sign as the input.

The VHDL code for the accumulator is presented in Program 3.4, where `data` is the new value to be added to the old value `sum`. The variable `prev` is the result of previous accumulation, which is used to check for overflow of the new result.

Program 3.4 *VHDL implementation of an accumulator.*

```vhdl
sum := prev+data; -- Calculate new sum
-- Check for negative overflow: if new sum is positive,
-- but previous value and data input were negative.
if sum(width_in-1) = '0' and prev(width_in-1) = '1'
                         and data(width_in-1) = '1' then
  -- Set sum to negative maximum
  sum := (width_in-1 => '1', others => '0');
-- Check for positive overflow: if new sum is negative,
-- but previous value and data input were positive.
elsif sum(width_in-1) = '1' and prev(width_in-1) = '0'
                            and data(width_in-1) = '0' then
  -- Set sum to positive maximum
  sum := (width_in-1 => '0', others => '1');
end if;
prev := sum; -- Set new previous value
-- Set result to the value of sum
result <= sum(width_in-1 downto width_in-width_out);
```

The Quartus II gives an estimation of the device space used by the design. BeMicroCV-A9 uses the 5CEFA9F23C8 FPGA, which has a total of 113 560 *adaptive logic module* (ALM). The whole FPGA implementation only uses 3.3 % of the available ALMs. Approximately 30 % of the total design logic utilization is spent on the control system synthesization, while the rest is spent on synthesizing the Nios II core. The ALM utilization for a single tap is 7.5 % of the total design. Even though the FPGA could theoretically fit tens or hundreds of taps, it would not be a viable design with this canceller architecture. Increasing the amount of taps on the canceller would also increase the physical dimensions and power consumption in addition to the hardware costs.

3.5 Digital-to-analog conversion

After the algorithm, the filter weights are still in two's complement representation. The wanted voltage range for the vector modulator is 0 V-3 V, which means that the weights have to be converted from two's complement to offset binary. This is done by inverting the MSB as shown in Table 3.2. A more intuitive way of thinking about this operation is just adding the absolute value of the negative maximum to the control value such that the result is offset to the unsigned two's complement range.

Table 3.2 *Comparison between two's complement and offset binary for a 3-bit number.*

Two's complement	Offset binary	Integer
011	111	3
010	110	2
001	101	1
000	100	0
111	011	-1
110	010	-2
101	001	-3
100	000	-4

The filter weights are truncated to 16-bits for the DAC. The 16-bit values become scaled from $-2^{15}...2^{15}-1$ to $0...2^{16}-1$ when switching to offset binary representation. This also maps the zero control value to 1.5 V, which is the null point of the vector modulator as shown in Figure 2.7. The conversion from the weight calculation result to actual control voltage is shown in Figure 3.12.

The VHDL code for changing the binary representation from two's complement to offset binary is presented in Program 3.5, where `data` and `result` are the input and output signals respectively and their number of bits is determined by `width`. The ** operator is exponentiation.

Program 3.5 *VHDL code for offsetting two's complement number for the DAC*

```
result <= unsigned(std_logic_vector(data))+2**(width-1)
```

The Analog Devices AD5676 [28] is a 16-bit, 8-channel DAC. It is controlled through SPI with an input clock frequency of up to 50 MHz. For simplicity the SPI clock is tied

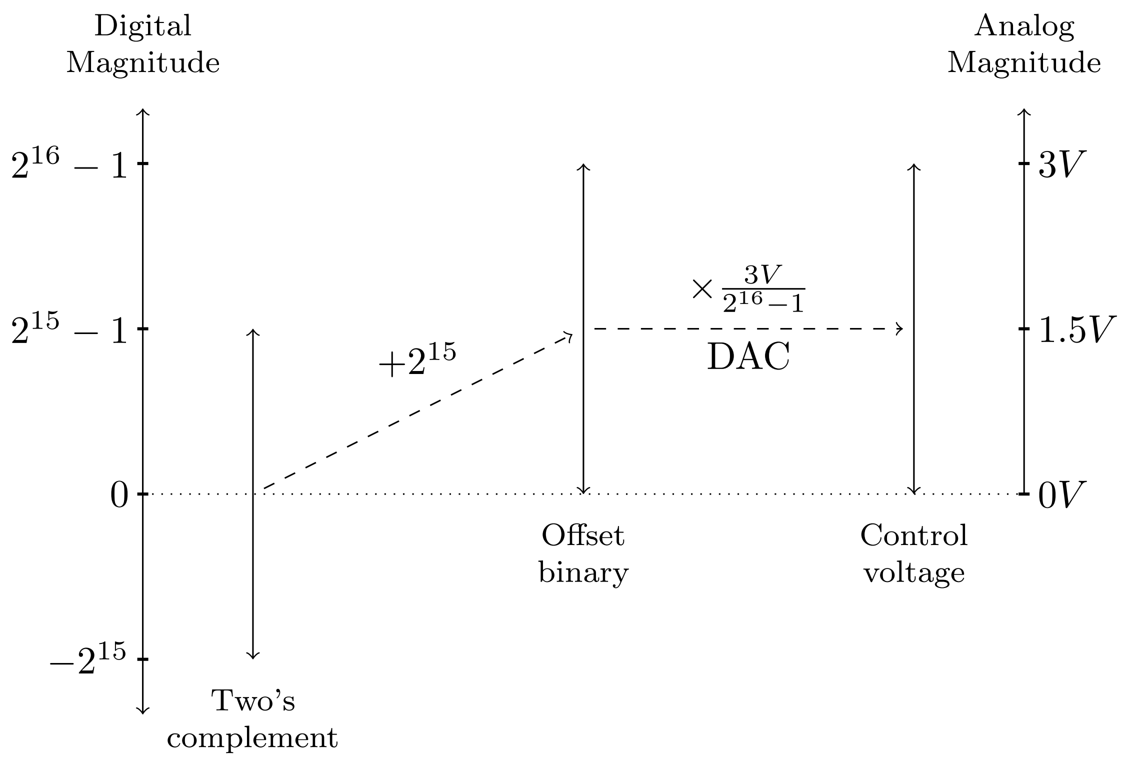


Figure 3.12 *Filter weight to control voltage conversion.*

to the same clock as FPGA fabric. A voltage reference of 1.5 V is provided by ISL21010 [29]. A gain setting of 2 is used, which means the the output voltage range is 0 V-3 V. On reset the DAC sets output at half-scale, 1.5 V.

Table 3.3 *AD5676 digital interface signals.*

Mnemonic	Description
SCLK	Serial clock input.
SDI	Serial data input.
$\overline{\text{SYNC}}$	Active low control input.
$\overline{\text{LDAC}}$	Active low load DAC input.
$\overline{\text{RESET}}$	Active low reset input.

The digital interface of the DAC consists of 5 signals, which are presented in Table 3.3. There is also a serial data output signal (SDO), which is left unconnected because neither readback nor daisy-chaining of multiple devices is required. The 40 MHz FPGA system clock is routed to SCLK, which indicates when the DAC samples the SDI signal. The data is sampled to the input register on falling edge of the SCLK. $\overline{\text{SYNC}}$ controls the synchronization of the input data frames. After $\overline{\text{SYNC}}$ is driven low, the data is transferred on the following 24 clock cycles. $\overline{\text{LDAC}}$, when low, passes the values from the input register to the DAC register. If $\overline{\text{LDAC}}$ is kept low, the channel outputs are updated immediately after $\overline{\text{SYNC}}$ is taken high. For this control system the $\overline{\text{LDAC}}$ is pulsed low after all of the channel inputs have been written, which updates all output channels simultaneously. $\overline{\text{RESET}}$ is connected to the same reset signal as the rest of the FPGA. The functional diagram is shown in Figure 3.13. The line over text implies that the signal is active low. In the code the active low signals are denoted by _N e.g. SYNC_N.

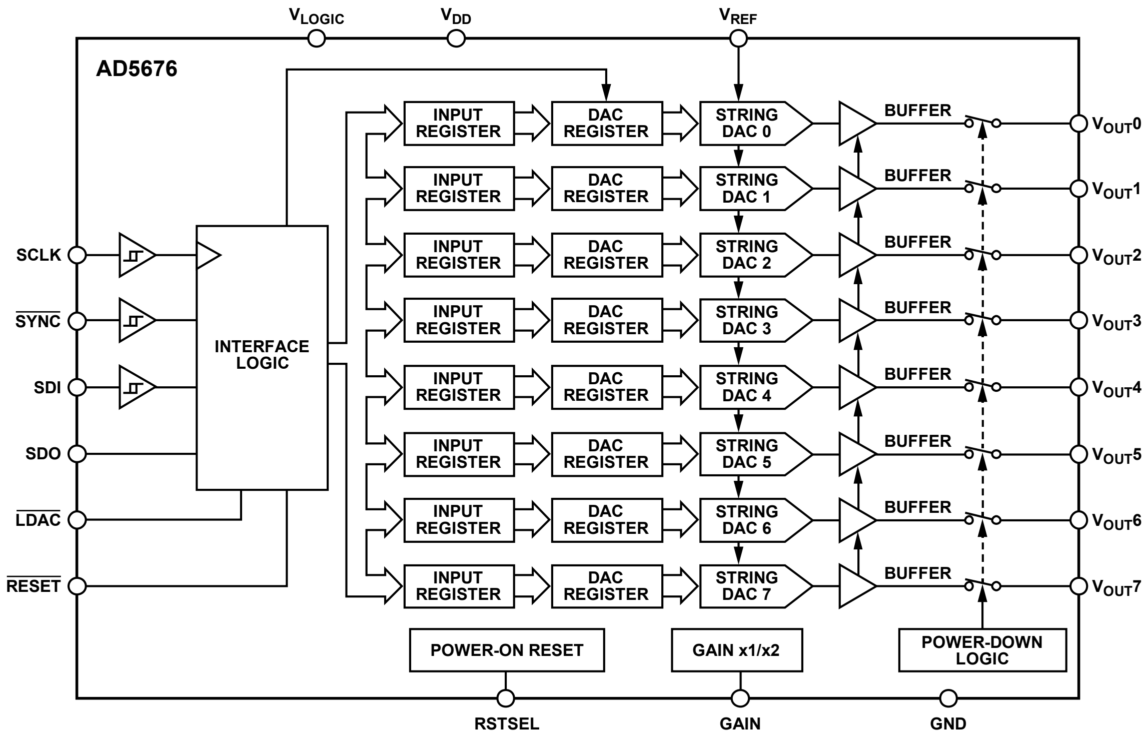


Figure 3.13 *DAC functional diagram [28].*

The state machine that controls the DAC is shown in the Figure 3.14. The first state is `Sample`, which samples the current values of the algorithm into registers. `Prepare`-state forms the bit vector, containing the serial data for a single DAC channel, that will be sent to the DAC. The bit vector consists of 4-bit command, 4-bit address and 16-bit data for a total of 24 bits. `Send`-state drives $\overline{\text{SYNC}}$ signal low and over the following 24 clock cycles the bit vector is written to SDO one bit at a time. After the last bit has been sent $\overline{\text{SYNC}}$ is driven back high and channel is switched and new bit vector is prepared. After all of the 6 channels have been written new control values are read into a register.

The simulated timing of the SPI interface is shown in Figure 3.15. The cursors are placed on the two consecutive falling edges of $\overline{\text{LDAC}}$. The time between the cursors denotes the time between two voltage updates. The 16-bit data values are set as do not care

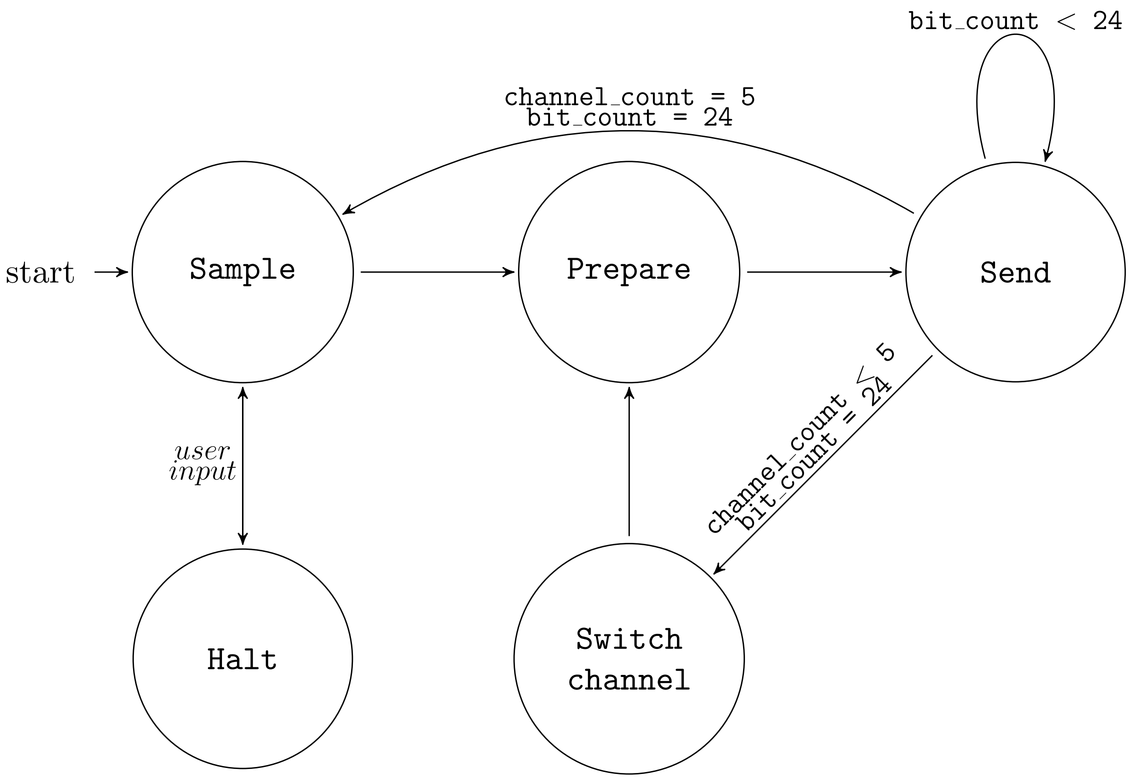


Figure 3.14 *SPI master state machine.*

states, which are shown as blue in the figure. It can be seen that the $\overline{\text{SYNC}}$ signal stays high for 3 clock cycles between the writes. This could potentially be reduced to just 1 clock cycle, but it would require combining states on the state machine, complicating the design. Overall the additional 2 clock cycles for each channel lead to a total overhead of 12 clock cycles. Note that in the figure SDO is the serial data output of the FPGA, which is connected to SDI of the DAC.

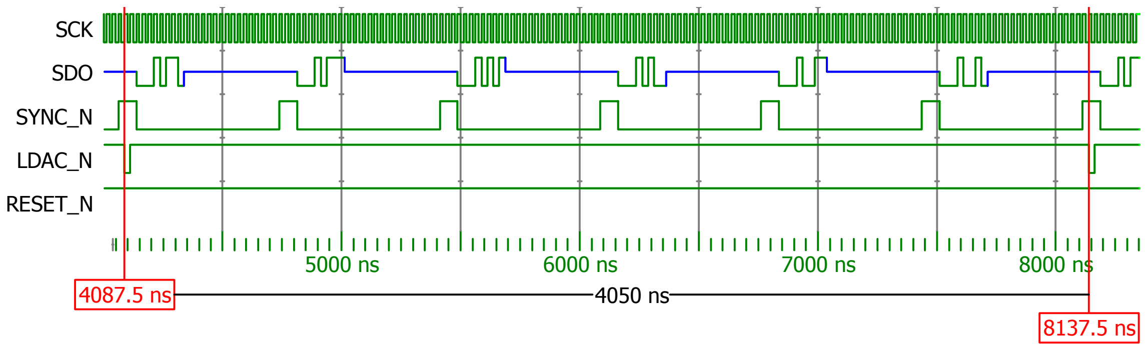


Figure 3.15 *Simulated timing of the SPI interface.*

The DAC is the largest factor in the delay of the control loop. Each output channel requires

24 clock cycles to write the value for the conversion and another clock cycle for the $\overline{\text{SYNC}}$ signal. Six channels are needed to cover I and Q controls for 3 taps which means that a total of 150 clock cycles are required to write the data to the DAC. In addition to the data transfer, there is also the 12 clock cycle overhead from the state machine running the SPI for a total of 162 clock cycles per update. The rate, at which the algorithm processes the baseband samples, is lowered to match the update rate of the control voltages. This is done by sending an enable signal to the accumulator once during the 162 clock cycles, or 4.05 μs. This lowers the accumulation rate of the algorithm to 247 kHz, equal to the update rate of the control voltages. The rate is fast considering the speed at which a human, for example, is moving near the antenna. An object traveling at 10 m/s, or 36 km/h, would only move 0.0405 mm between two weight updates. This rate could further be improved by using multiple parallel DACs, or a DAC with a serial interface faster than 40 MHz SPI.

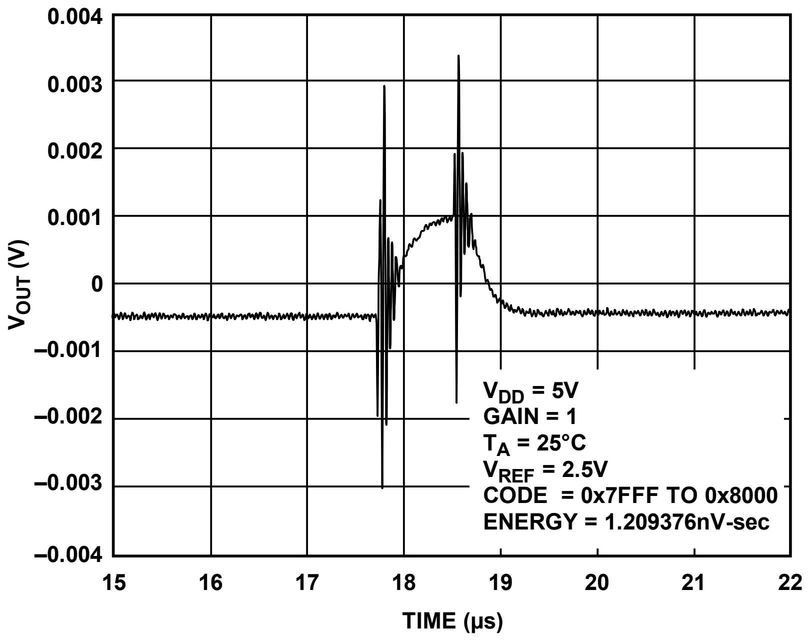


Figure 3.16 *DAC Glitch Impulse [28].*

When a DAC channel is updated, there is a glitch impulse of up to 6 $\text{mV}_{\text{p-p}}$ and a duration of 1 μs. The glitch impulse is presented in Figure 3.16. The impulse affects the vector modulator and therefore the cancellation and the feedback signal. The accumulation enable signal is timed after the glitch to prevent any effect on the convergence of the algorithm.

3.6 User interface

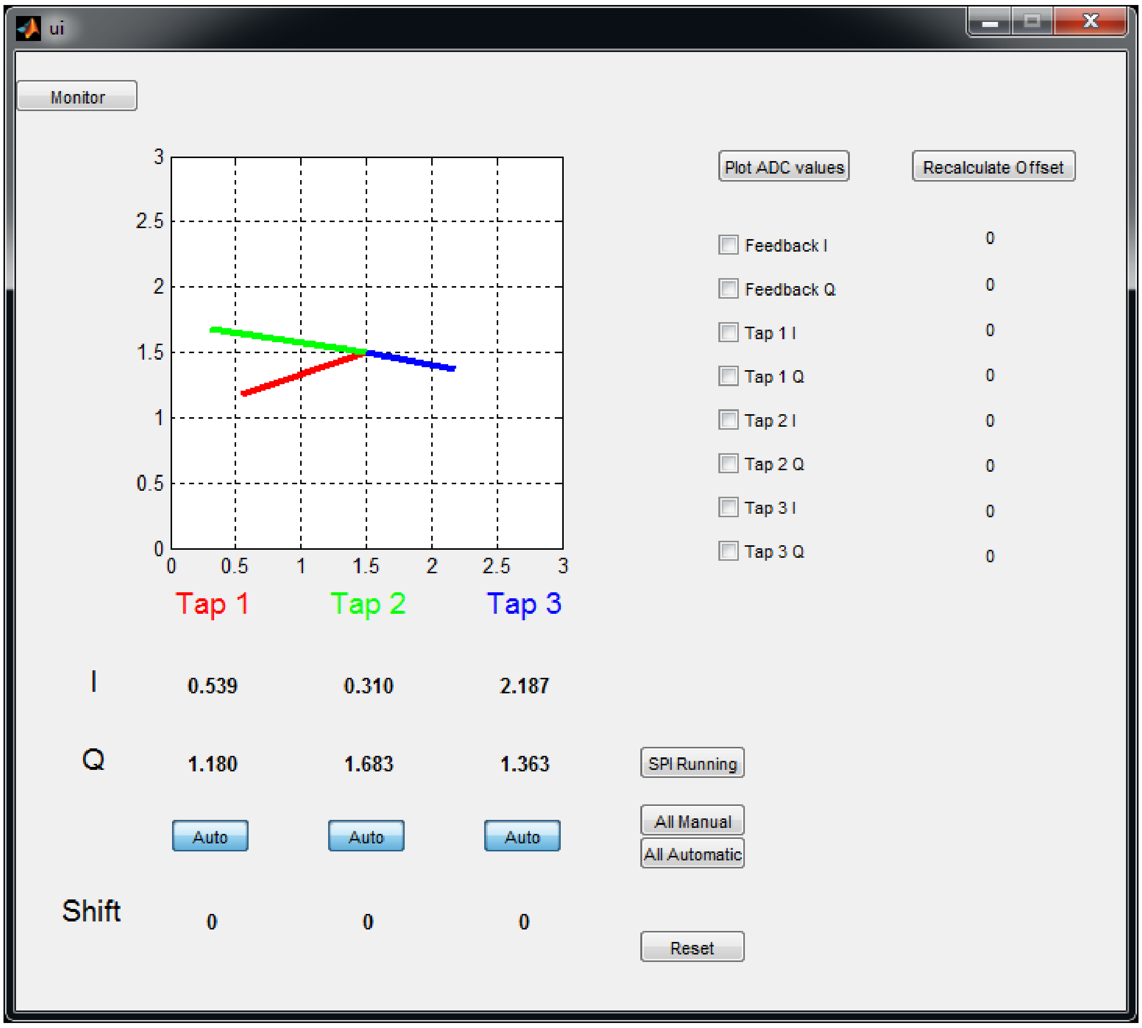
The *user interface* (UI) was created to allow the algorithm parameters to be adjusted on the fly without reprogramming the FPGA. Creating the interface directly on the FPGA with VHDL would be possible, but time consuming. The interface is based on passing data between the FPGA fabric and the computer through a soft microcontroller NIOS II. A microcontroller allows the interface software to be developed much faster using a high level programming languages. A soft microcontroller can be implemented on the FPGA with logic synthesis, which simplifies the PCB design process because a discrete microcontroller is no longer needed. Also the number of necessary I/O pins is reduced significantly. The system is controlled through UART. A serial terminal program with a text based interface can be used with keyboard alone. A graphical UI was also created in *MATLAB*, which allows more intuitive control.

The Nios II has access to the FPGA fabric through custom instructions. The custom instructions can be used to improve performance of software algorithms with hardware acceleration, but in this UI they were used to control the variables of the algorithm and reading the ADC and DAC values back to the user for debugging purposes.

The UI contains the following features:

- Toggling between automatic and manual control for each tap separately
- Manually setting the vector modulator control voltages
- Adjusting the step-sizes for each tap when on automatic control
- Plot 128 consecutive samples from the ADC channels
- Plot 65536 consecutive DAC control values
- Display and reset the input signal offset calculations
- Display the control voltages in the IQ-plane in real time

The software for Nios II can be programmed in C even though the rest of the system uses VHDL. The soft microcontrollers are not limited to fixed amount of peripherals like traditional microcontrollers. The designer can select the amount of memory and required peripherals based on the systems requirements. Only what is needed will be



	Tap 1	Tap 2	Tap 3
I	0.539	0.310	2.187
Q	1.180	1.683	1.363
Shift	0	0	0

Feedback I	0
Feedback Q	0
Tap 1 I	0
Tap 1 Q	0
Tap 2 I	0
Tap 2 Q	0
Tap 3 I	0
Tap 3 Q	0

***Figure* 3.17** *UI in MATLAB.*

synthesized, which leads to more efficient utilization of the logic resources. The soft microcontrollers can be changed during the design process to match the current needs. For a traditional hard microcontrollers increasing the performance is only possible through software optimizations and increasing clock rate, but increasing the amount of IO, for example, is impossible. The Nios II runs on the same 40 MHz clock as the algorithm.

The interface is also an important debugging tool. The values of the ADC can be read and plotted with MATLAB to see the waveform of the feedback and tap baseband signals. The maximum data rate of the 8 LVDS channels is 4.8 Gbps, which is over four orders of magnitude higher than the baud rate of the UART. To capture a segment of the incoming signal a *first-in-first-out* (FIFO) buffer was added. The buffer holds 128 samples which translates to 3.2 µs segment at 40 MSPS sampling rate. This is enough to capture a sine wave test signal to verify the correct analog-to-digital conversion and deserialization. Correspondingly a FIFO buffer was added for the DAC control values which holds 65536 samples. Unlike the ADC FIFO the DAC FIFO is not written on every clock cycle, but only when the control values are updated. This allows the DAC FIFO to hold samples from a much larger time window of 265 ms. The buffers work in parallel with the algorithm, so it has no effect on the latency of the control.

When the UI is opened, the communication between the Nios II and PC is initialized by creating a serial port object `serial`. The serial port, baud rate and terminator character are given as parameters. The serial port object is stored into a data structure `handles`, which is used to pass data between different functions. After creating the serial port object, it is connected to the device. The code for the serial port initialization is presented in Program 3.6.

Program 3.6 *MATLAB code for creating a serial port object and opening a connection.*

```
handles.serial = serial('COM5','BaudRate',115200,'Terminator','LF');
fopen(handles.serial);
```

The communication port is a virtual one, allowing access to *universal serial bus* (USB) device as if it were a standard communication port. The USB-UART interface is provided by a FT232R chip [30].

Pressing a button on the UI causes a callback function, specific to the button, to be executed. For example, the *All Automatic*-button shown in Figure 3.17 sends a string to Nios II containing the command. The command is sent by printing the command string to serial port object. The string is terminated with carriage return '\r'. The command transmission is presented in Program 3.7.

Program 3.7 *MATLAB code for sending the command through serial port.*

```
fprintf(handles.serial,'%s\r','All Auto');
```

The string is transfered through UART one character at a time. Every character causes an *interrupt service routine* (ISR), presented in Program 3.8, to be called. The ISR reads each character into a buffer until a termination character '\r' is receiver. After the termination character has been received the string contained in the buffer is terminated with null '\0' so it can be treated as null terminated string allowing use of string comparison functions to interpret the incoming commands.

Program 3.8 *Nios II UART interrupt service routine.*

```c
void UART_RECEIVE_ISR(void* context){
  // Read the received character from register
  char data = IORD_ALTERA_AVALON_UART_RXDATA(UART_BASE);

  // The command inputs end with carriage return '\r'
  // After receiver '\r' the command is ready to be interpreted
  if(data == '\r'){
    // Add null to the end of the received characters
    // so we can handle the buffer as null terminated string (char*)
    rxBuffer[bufferCount] = '\0';
    // Next data is read to the start of the buffer
    bufferCount = 0;
    // Buffer contents are ready to be interpreted
    bufferReady = true;
    return;
  } else {
    // Add the received character to the buffer
    rxBuffer[bufferCount] = data;
    bufferEmpty = false;
    // Increment buffer count
    ++bufferCount;
  }
}
```

The contents of the buffer can now be compared to predetermined commands in the Nios II program memory. This is presented in Program 3.9. If the null terminated string in the buffer matches the command, the specified custom instructions will be called. When the command is to set all taps on automatic control, DAC FIFO buffer will be flushed. The DAC buffer has been programmed to start capturing samples when it is empty and to stop capturing when it is full. This allows to capture the convergence of the algorithm from the DAC control values. In this implementation the custom instructions have two parameters: instruction number and data. In the case of enabling automatic control, the data parameter can simply be ignored because the instruction number alone is enough to set the correct signals on the FPGA.

Program 3.9 *Interpreting the command in Nios II.*

```
// Compare the contents of the buffer to the command
if(strncmp(RxBuffer,"All Auto",8) == 0){
  // Clear DAC fifo to start capturing samples immediately
  ALT_CI_CUSTOM_INSTRUCTION_0(65,0);
  // Set all channels to automatic control
  ALT_CI_CUSTOM_INSTRUCTION_0(40,0);
}
```

On the FPGA the control method is determined by a 3-bit bus named `manual_control`. Each of the bits correspond to a specific tap. The custom instruction number 40 sets all channels to automatic control by setting the bus to all zeros. Conversely the custom instruction to enable manual control sets the bus to all ones. The bus is routed to the SPI master, whose code is available in appendix B. The accumulators for each tap also use the information of the control method, to reset the accumulator sum when the control is set to manual. This sets the control values to 0 in order to allow the algorithm to start from the theoretical vector modulator null point when automatic control is enabled. The custom instruction in presented in Program 3.10.

Program 3.10 *VHDL implementation of custom instruction number 40.*

```
-- Check if the instruction number n is 40
if clk_en = '1' and n = "00101000" then
      -- Set all taps to automatic
      manual_control <= (others => '0');
end if;
```

All of the UI features follow this same basic principle of passing a command from MATLAB to Nios through UART and from Nios to FPGA fabric through custom instructions.

4. MEASUREMENTS AND RESULTS

This chapter presents the measurement results of the analog canceller and the control system. First the measurement setup is briefly explained. The presented cancellation results are analyzed along with the adaptivity and convergence of the control system. Finally, the results are compared with existing analog canceller designs.

4.1 Measurement setup

The measurements were carried out using a *vector signal transceiver* (VST). The canceller RF input signal was amplified using a PA with a gain of 24 dB. The delay cable lengths were 35 cm for the shorter delay and 1 m for the longer delay. The velocity factor of the coaxial cables used was 0.66, which determines speed of the signal in the cable in relation to the speed of light. Each meter of cable is equal to $\frac{1\,\mathrm{m}}{299792458\,\frac{m}{s}} \times 0.66 = 5.05\,\mathrm{ns}$ of delay. The fixed delay tap is aligned with the circulator leakage, while the second tap is aligned with the antenna reflection. The measurement environment was standard lab environment. The block diagram and a photograph of the measurement setup are presented in Figures 4.1 and 4.2 respectively.

- Vector signal transceiver: National Instruments PXIe-5645r
- Oscilloscope: Keysight InfiniiVision MSO-X 4104a
- LO-signal generator: Hewlett Packard E4437B
- PA: Texas Instruments CC2595
- Circulator: JCC2300T2500S6
- LNA: HD Communications Corp. HD24089
- Power supplies: Hewlett Packard E3631A and Aim-TTi PL303QMD-P

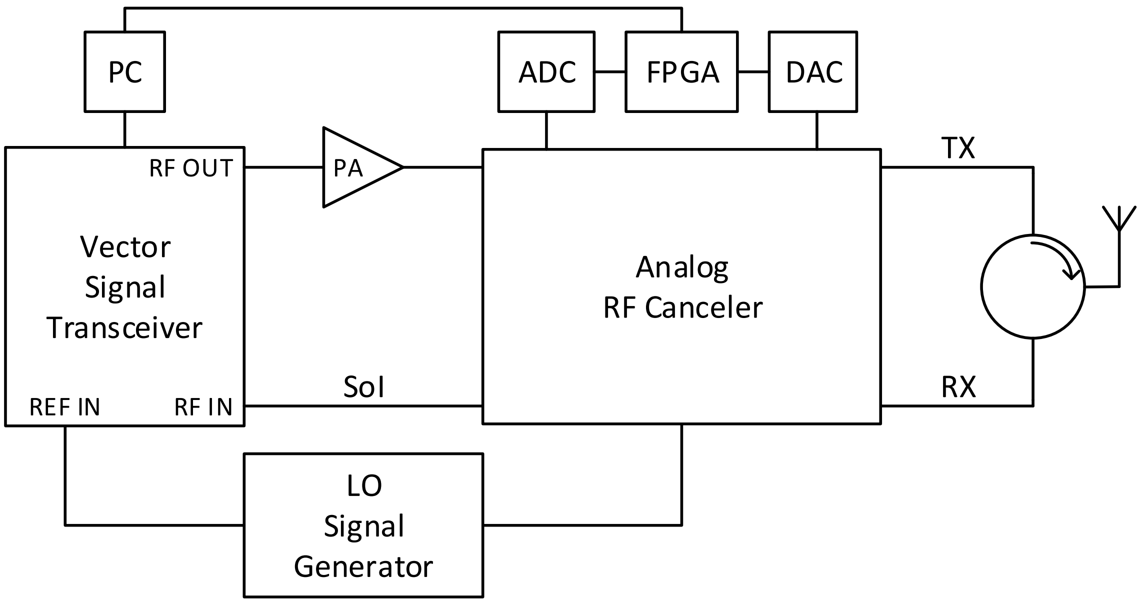


Figure 4.1 *The measurement setup block diagram.*

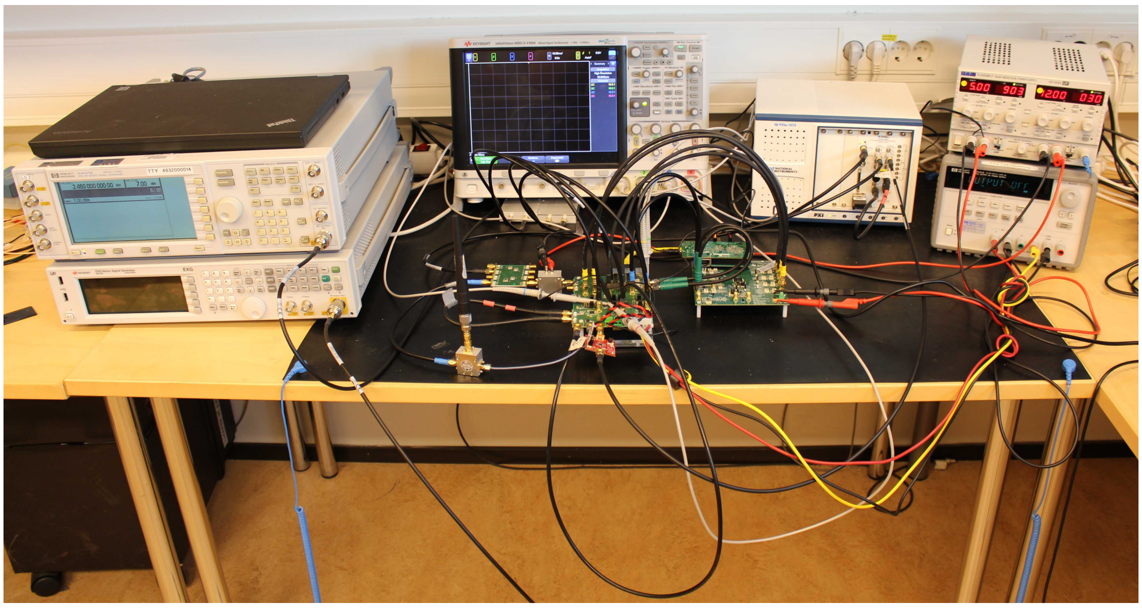


Figure 4.2 *The measurement setup photograph.*

The power consumption of the measurement setup is presented in Table 4.1. Majority of the power is consumed by the downconverters. Each of the 4 downconverters consumes roughly 300 mA with 5 V supply voltage [12]. Additionally, the PA is powered from the same 5 V supply as the canceller. The 8 V supply powers the 3 vector modulators, each of which typically consumes 93 mA [14].

Table 4.1 *Power consumption of the measurement setup.*

Device	**Voltage**	**Current**	**Power**
Canceller	5 V	1300 mA	6.5 W
Canceller	8 V	300 mA	2.4 W
ADC	5 V	400 mA	2 W
FPGA	5 V	400 mA	2 W
LNA	12 V	20 mA	240 mW
DAC	5 V	10 mA	40 mW

4.2 Cancellation

The cancellation with 20 MHz, 40 MHz and 80 MHz signal bandwidths are presented in Figures 4.3, 4.4 and 4.5 respectively. Each figure shows the power of the PA output, circulator input, and the RX signal before and after applying the cancellation signal. It should be noted that the circulator output and the RX signal cannot be measured simultaneously. This means that the ratio between intrinsic attenuation and active cancellation may vary during the measurements. However, the circulator input power is not affected, which means that the total cancellation is the most accurate representation of how much the power of the SI has been reduced. In each figure a small spike can be seen on the center frequency due to LO leakage. The LO frequency for these measurements was 2.46 GHz. The signal powers presented in figures 4.3, 4.4 and 4.5 are summarized in Table 4.2.

Table 4.2 *The signal power in each stage of the canceller.*

Signal bandwidth	**TX power**	**RX power (canceller disabled)**	**RX power (canceller enabled)**
20 MHz	6.4 dBm	−13.5 dBm	−61.8 dBm
40 MHz	8.0 dBm	−14.1 dBm	−58.6 dBm
80 MHz	8.0 dBm	−14.2 dBm	−55.2 dBm

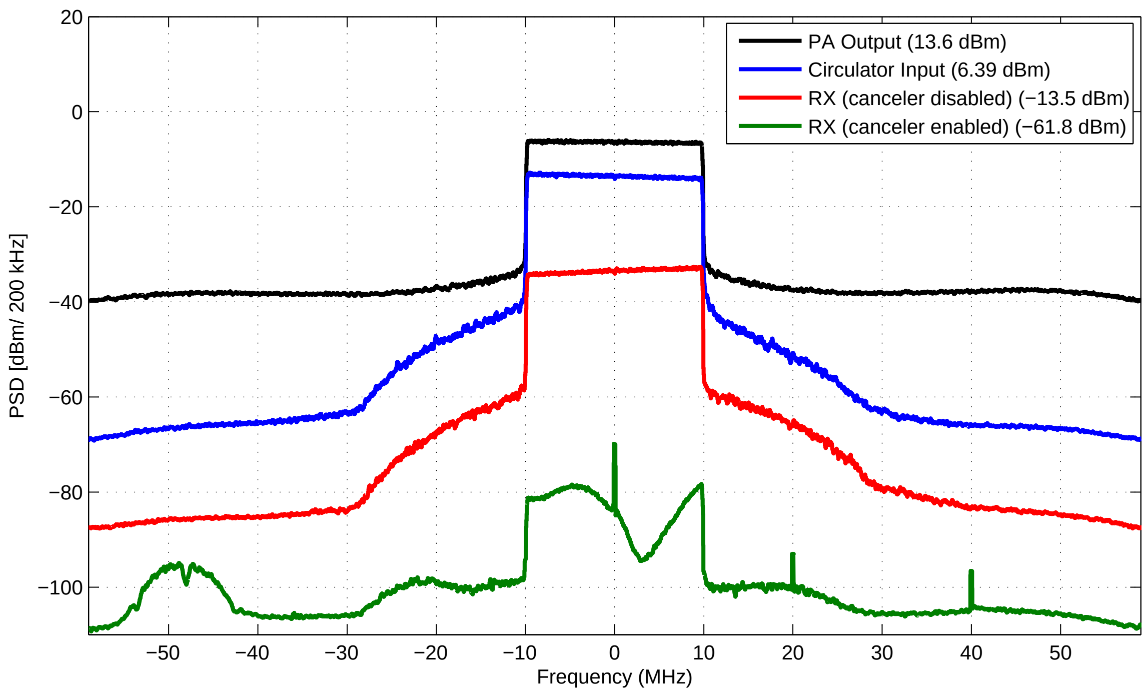


Figure 4.3 *Cancellation with 20 MHz bandwidth.*

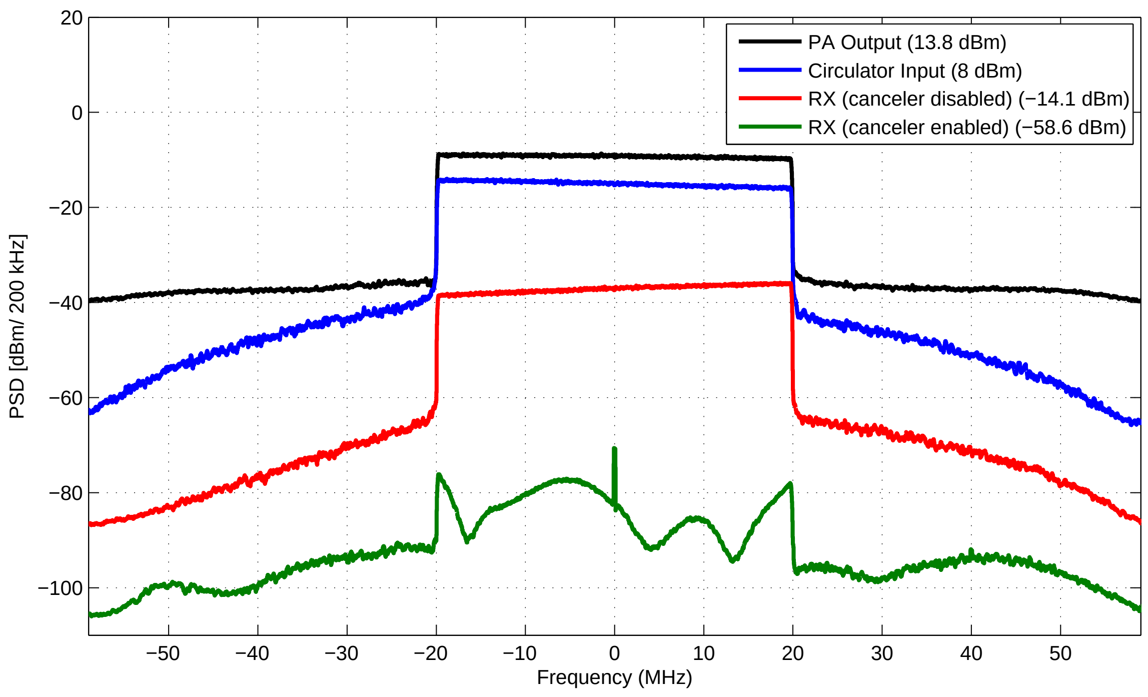


Figure 4.4 *Cancellation with 40 MHz bandwidth.*

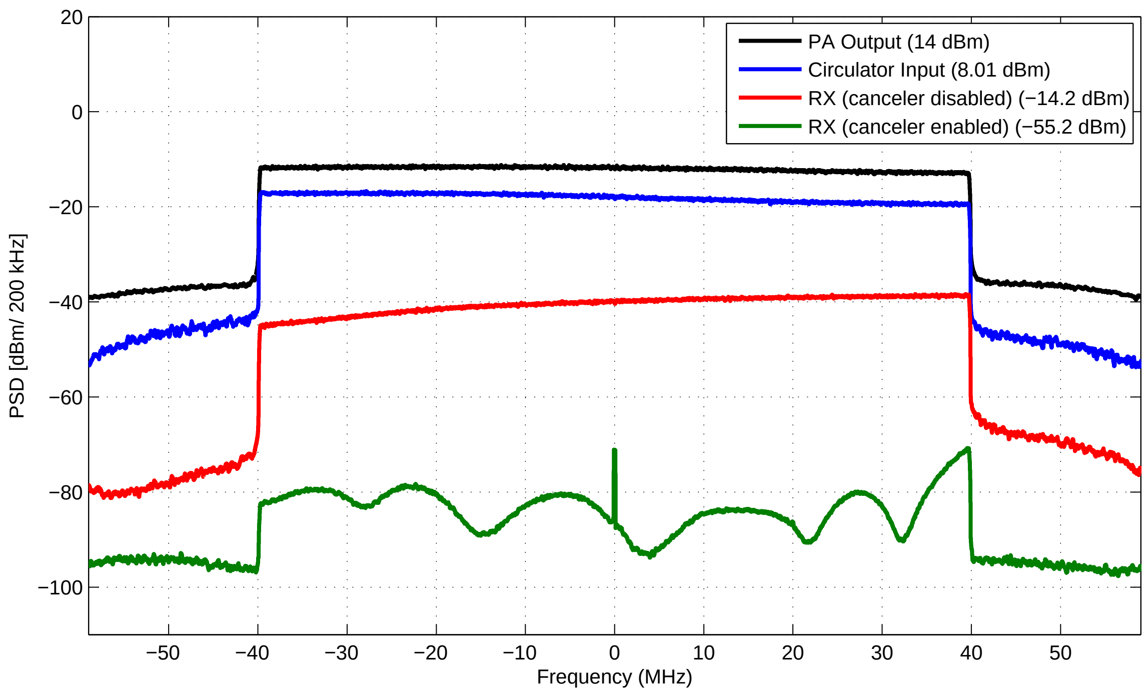


Figure 4.5 *Cancellation with 80 MHz bandwidth.*

The amount of intrinsic, active and total cancellation with different bandwidths are presented in Table 4.3. Intrinsic attenuation refers to the reduction of SI power without active cancellation and it can be calculated as the difference of circulator input power and RX power when the canceller is disabled. It is strongly dependent on the circulator and antenna used in the measurements. Active cancellation describes how much further the power can be reduced by applying the cancellation signal and it can be calculated as the difference of RX power with canceller disabled and enabled. The total cancellation is a sum of intrinsic attenuation and active cancellation. It can be seen that increasing the signal bandwidth reduces the amount of active cancellation, and therefore the total cancellation achieved.

Table 4.3 *Maximum cancellation achieved with different signal bandwidths.*

Signal bandwidth	**Intrinsic attenuation**	**Active cancellation**	**Total cancellation**
20 MHz	19.9 dB	48.3 dB	68.2 dB
40 MHz	22.1 dB	44.5 dB	66.6 dB
80 MHz	22.2 dB	41.0 dB	63.2 dB

4.3 Adaptivity and convergence

The convergence of the control voltages is shown in Figure 4.6. At the start the power of the feedback signal is very high, thus the steps of the algorithm are very large. As the cancellation improves the feedback power becomes lower. The power of the tap signals remain constant regardless of the cancellation. The convergence of the gain and phase of all taps are shown in Figure 4.7. During the first 5 to 10 milliseconds, all three taps are very close to each other. This is to be expected as the only difference between the taps is the amount of delay. The differences in gain and phase between the taps increase slowly over thousands of iterations. AGC could make the convergence significantly faster, because the individual steps taken by the algorithm would not be limited by the power of the feedback signal. Figures 4.6 and 4.7 only show the convergence for this specific setup, moving the antenna or changing components such as the circulator would cause a convergence to different values.

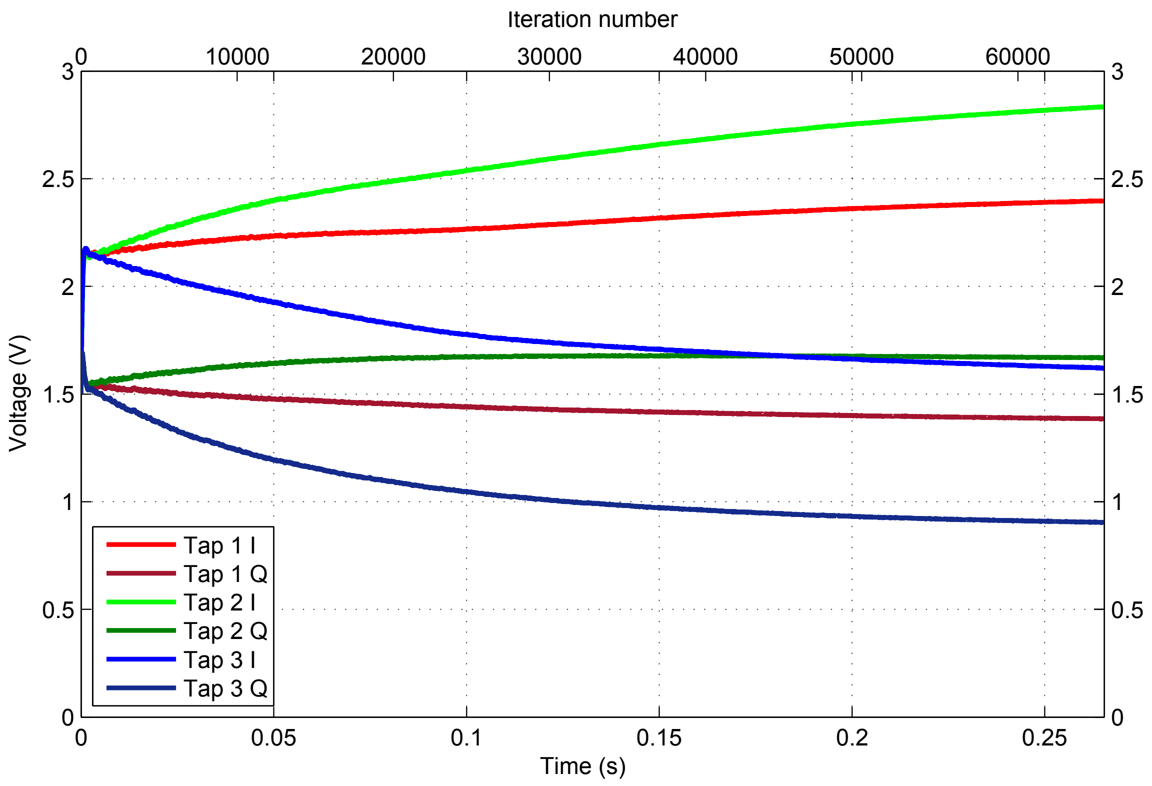


Figure 4.6 *The convergence of the control voltages for all taps.*

The behavior of the feedback signal power during the first 5 ms of convergence was measured using an oscilloscope. Two probes were connected to the feedback I and Q signals and another two to the control voltages of the fixed delay tap. The control voltages were

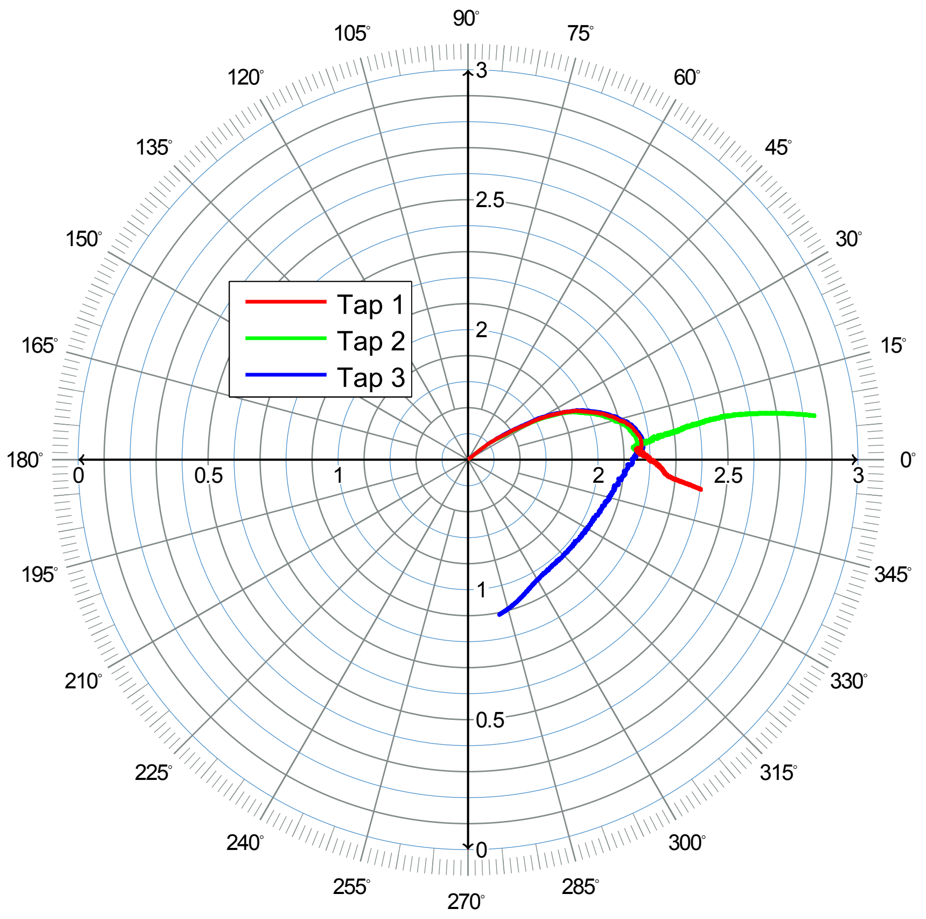


Figure 4.7 *The convergence of the control voltages for all taps in IQ-plane. The units on the X and Y axes are volts for the I and Q control respectively. The figure also displays the phase change for each tap. The gain control can be determined by the distance from origin according to the equation 2.2.*

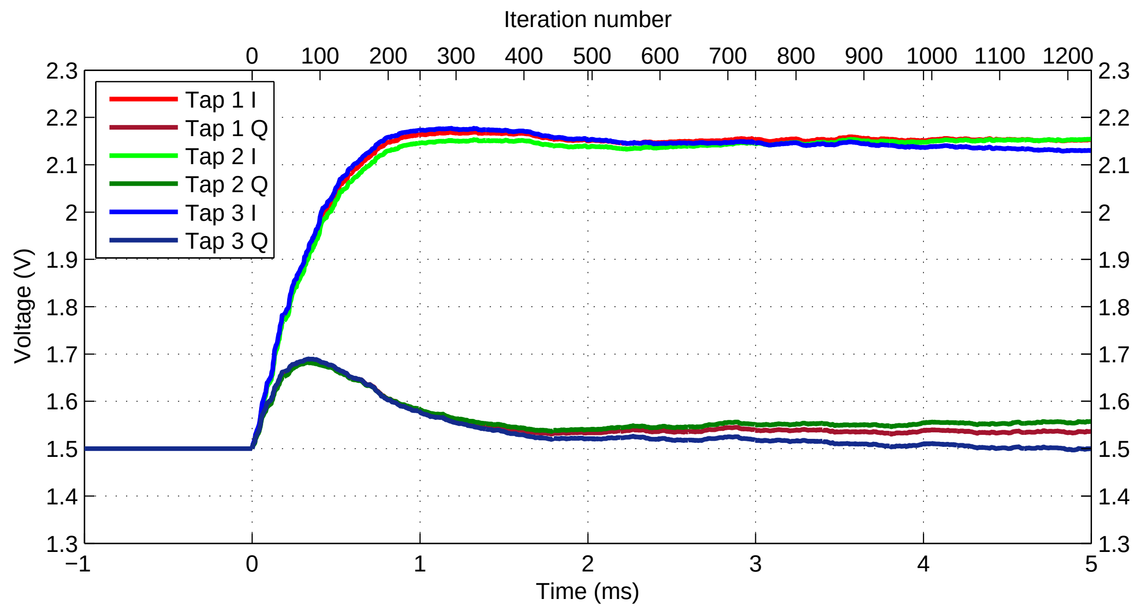
set to 1.5 V and step-size was set to 1 (no bit-shifting). The oscilloscope was set to trigger from one of the control voltages when it's level crossed 1.6 V. The fixed delay tap was chosen for the trigger because it targets the most static source of SI, circulator leakage, which was deemed the most predictable. FIFO buffer on the FPGA captures 65536 consecutive samples of the DAC control values from the point where the control is switched to automatic and the algorithm is enabled.

The oscilloscope's sampling rate depends on the time/div setting such that the memory is filled with the data of the current window. Increasing the time scale lowers the sampling rate and vice versa. In addition, the sampling rate is dependent on the enabled channels. If channels 1 and 2 or 3 and 4 are used simultaneously, the sampling rate is halved. For the measurement a total of three channels are needed; two channels for the feedback signals and one channel for the control voltage sensitive trigger. Adding the fourth channel will not cause any further performance loss. For these reasons only a fairly short segment of the voltages can be displayed, while still being able to properly capture the 10 MHz I and Q components of the feedback signal. Due to the noise level of the oscilloscope the signal cannot be measured with the same accuracy as the VST in the cancellation measurements, but it is enough to show the rapid convergence in the beginning. The measurement accuracy could be improved by using coaxial cables instead of the probes, but it is not possible to use them at the same time as the feedback signal is connected to the control system ADC. Splitting the feedback signal would cause the the ADC to hit noise floor before the maximum cancellation has been achieved.

As mentioned in section 3.5, each iteration takes 4.05 µs which means that each millisecond corresponds to 247 iterations. It can be seen from Figure 4.8 that in just 1.2 ms from the moment the automatic control is enabled, the SI is already reduced by 20 dB. As mentioned earlier, the magnitude of the individual steps taken by the algorithm are relative to the power of the feedback signal. For this reason, the algorithm convergence becomes significantly slower as 20 dB of cancellation means that less than 1 % of the feedback signal power is remaining. Convergence to the maximum cancellation can take as long as 250 ms. However, once the maximum cancellation is reached, the tracking capabilities of the control system are able to maintain the cancellation within 5 dB of the maximum even when the antenna is disturbed with metallic objects. This behavior is shown on a video available at:

- `http://www.tut.fi/full-duplex/RFCancDemo4.mp4`

The video demonstrates the behavior of the RX signal power with the control system enabled and disabled while the antenna is being disturbed. The video also shows the behavior of the control voltages as seen on the screen of an oscilloscope.



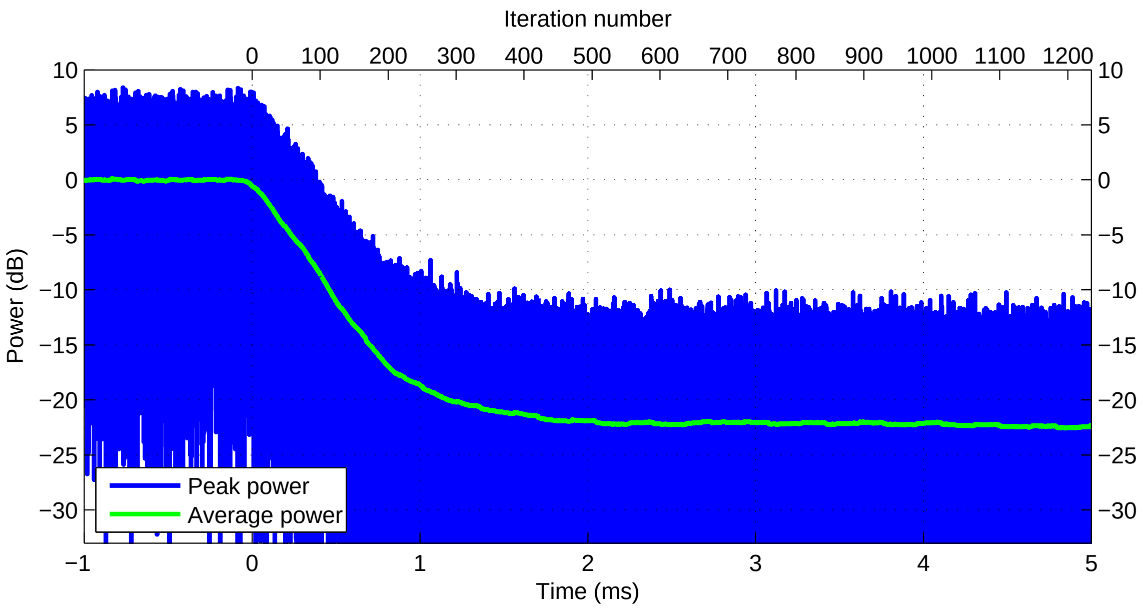


Figure 4.8 *Top: The convergence of the control voltages over 5 ms. Bottom: The instantaneous and average power of the feedback signal over 5 ms.*

4.4 Comparison

The cancellation results achieved by other existing canceller designs are compared to this design in Table 4.4. The improvement of the active cancellation compared to the previous revision of the canceller can be credited to the additional tap.

Table 4.4 *Comparison between different designs in lab conditions with 20 MHz bandwidth.*

Design	Number of taps	Intrinsic attenuation	Active cancellation	Total cancellation
This design (rev. 3)	3	20 dB	48 dB	68 dB
Previous revision	2	22 dB	34 dB	56 dB
MIT	4	56 dB	22 dB	78 dB
Stanford	16	15 dB	57 dB	72 dB

The MIT design [2] uses a 4-tap canceller. While the amount of active cancellation is fairly low, they are able to achieve nearly 80 dB of total cancellation by using a high isolation antenna. Each tap contains a delay line, a variable attenuator and a phase shifter, whose weights are tuned using a *dithered linear search* (DLS) algorithm. DLS is a type of gradient descent algorithm, where the weights are calculated using only the *received signal strength indicator* (RSSI). They report that the algorithm achieves sufficient cancellation in 500 µs. To find multiple local minima of the RSSI, they use multiple different initialization values for the algorithm. However, they do not mention any real-time tracking capabilities in dynamic environment. Reinitializing the algorithm to find new local minima will cause the cancellation to be poor in the beginning of the tuning, so it cannot be used to track and maintain good cancellation continuously. The benefit of DLS is that the filter weights can be calculated without any knowledge of the TX signal.

The Stanford design [3] achieves an impressive 57 dB of active cancellation. They use a single antenna with a circulator, which provides 15 dB of isolation. Their canceller contains a total of 16 taps with 8 taps centered around the circulator leakage and another 8 around the antenna reflection. Each tap consists of a delay line and a variable attenuator, but they do not have any phase control. They measure the frequency response of the antenna and circulator and calculate the attenuator weights offline. The weights are fine tuned with a gradient descent algorithm. The channel has to be free of transmission during the weight calculation, which they report to take less than 1 ms. They also report that the weights need to be recalculated once every 100 ms, but they do not mention how this number was obtained.

5. CONCLUSION

This thesis introduced a digital control for a 3-tap analog RF SI canceller. The goal was to show that digital control system is capable of fast convergence and stable cancellation. It is an important step towards commercial full-duplex applications. The control system was implemented using a high-speed ADC, an FPGA and a DAC. Delayed copies of the TX signal were used to create an adaptive filter, which tunes the canceller. The hardware has not been optimized for size or power consumption as the purpose was to provide proof of concept for a digital control system.

Using the digital control system the canceller was able to provide over 40 dB of active cancellation over a wide bandwidth of up to 80 MHz. The results achieved compare well with other similar systems reported in the academia [2, 3].

The adaptivity of the control system is vital to the operation of an RF canceller. As the environment of the antenna changes, the SI is also affected. LMS algorithm was used to allow the filter weights to be adjusted continuously while the RF signal is enabled. This means that once the algorithm has converged, it will continue to track the filter weights such that the cancellation remains stable. The filter weight calculation methods mentioned in other designs [2, 3] will lack proper cancellation during the calculation. A video was presented demonstrating the tracking capabilities of this control system.

The FPGA implementation requires very low amount of logic resources allowing low-cost FPGAs to be used or it could be added to an existing design with ease. In addition to the small resource utilization, the control system has very low latency of 425 ns excluding the DAC. In the current design the DAC is a significant bottleneck that increases the latency and update rate to 4.05 µs and 247 kHz respectively. This is still very fast considering that anything moving disturbing the antenna will move relatively slow compared to the time between two updates.

A conference paper was written regarding the digitally controlled canceller presented in this thesis [6]. The cancellation results, obtained using the digital control system, were also utilized in another scientific article [7], which also includes digital cancellation. While the results are already very promising there is still a lot of research and development to be done before commercial products are able to fully utilize full-duplex.

5.1 Future work

The current control system updates the control values at a fairly low rate due to the limited data rate of the DAC SPI. Replacing the current DAC with multiple DACs in parallel or a multichannel DAC with a parallel digital interface would allow the control system to benefit from the full performance of the FPGA. Higher update rate would allow even faster convergence and more stable cancellation in a dynamic environment.

The current design uses a coupler to capture the feedback signal before the receiver. Running the control algorithm on the same device as the receiver would eliminate the need for the coupler and the leftover feedback chain, which would reduce power consumption, PCB area and component expenses. This would not be possible with an analog control system. However, using the external feedback chain has its own benefits. It allows the canceller and control system to be added to a transceiver system without any modifications to the receiver.

Ultimately, for a consumer product, the entire control system could be miniaturized into a single mixed-signal integrated circuit. This would require significant development effort, but for a viable full-duplex system it is necessary to minimize the physical size and power consumption. The same need for miniaturization also applies to the canceller itself.

REFERENCES

[1] T. Huusari, Y.-S. Choi, P. Liikkanen, D. Korpi, S. Talwar, and M. Valkama, “Wide-band self-adaptive RF cancellation circuit for full-duplex radio: Operating principle and measurements,” in *Proc. IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–7.

[2] K. Kolodziej, J. McMichael, and B. Perry, “Multi-tap RF canceller for in-band full-duplex wireless communications,” *IEEE Transactions on Wireless Communications*, Mar. 2016.

[3] D. Bharadia, E. McMilin, and S. Katti, “Full Duplex Radios,” *Proc. SIGCOMM’13*, Aug. 2013.

[4] D. Korpi, Y.-S. Choi, T. Huusari, S. Anttila, L. Talwar, and M. Valkama, “Adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio: algorithms and RF measurements,” in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2015.

[5] Y.-S. Choi and H. Shirani-Mehr, “Simultaneous transmission and reception: Algorithm, design and system level performance,” *IEEE Transactions on Wireless Communications*, vol. 12, no. 12, pp. 5992–6010, Dec. 2013.

[6] J. Tamminen, M. Turunen, D. Korpi, T. Huusari, Y.-S. Choi, S. Talwar, and M. Valkama, “Digitally-controlled RF self-interference canceller for full-duplex radios,” in *Proc. European Signal Processing Conference (EUSIPCO)*, Aug. 2016.

[7] D. Korpi, J. Tamminen, M. Turunen, T. Huusari, Y.-S. Choi, L. Anttila, S. Talwar, and M. Valkama, “Full-duplex mobile device – pushing the limits,” *IEEE Communications Magazine*, submitted for review, 2016.

[8] T. Huusari, “Analog RF cancellation of self interference in full-duplex transceivers,” Master’s thesis, Tampere University of Technology, Tampere, Finland, May 2015.

[9] C. Shannon, “Communication in the presence of noise,” in *Proc. IRE*, Jan. 1949.

[10] D. Korpi, T. Riihonen, V. Syrjälä, L. Anttila, M. Valkama, and R. Wichman, “Full-duplex transceiver system calculations: analysis of ADC and linearity challenges,” *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3821–3836, Jul. 2014.

[11] *Surface Mount Power Splitter/Combiner*, Mini-Circuits, rEV.C. [Online]. Available: https://www.minicircuits.com/pdfs/BP4U+.pdf

[12] *High-Dynamic-Range, Direct Up-/Downconversion 1500MHz to 2500MHz Quadrature Mod/Demod*, Maxim, 2012, rev. 1. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX2023.pdf

[13] *Hittites Vector Modulators*, Hittite Microwave, 2008, v00.0608. [Online]. Available: http://www.analog.com/media/en/technical-documentation/application-notes/Vector_Modulators.pdf

[14] *GaAs HBT VECTOR MODULATOR 1.8 - 2.7 GHz*, Hittite Microwave, v00.1007. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/hmc631.pdf

[15] *Ultra-Small Ceramic Power Splitter/Combiner*, Mini-Circuits, rEV. D. [Online]. Available: https://www.minicircuits.com/pdfs/SCN-3-28.pdf

[16] *BeMicro CV A9 FPGA Development Board Hardware Reference Guide*, Arrow, 2015, rev. 2015.05.04. [Online]. Available: http://www.alterawiki.com/uploads/f/f3/Hardware_Reference_Guide_BeMicro_CV_A9_1.2.pdf

[17] *BeMicro CV FPGA Development Board Hardware Reference Guide*, Arrow, 2013, rev. 1.1. [Online]. Available: http://www.alterawiki.com/uploads/4/41/Hardware_Reference_Guide_for_BeMicro_CV_1.1.pdf

[18] *ADS5295, 8-channel, Analog-to-Digital Converter Evaluation Module*, Texas Instruments, 2012. [Online]. Available: http://www.ti.com/lit/pdf/slau442

[19] *12-bit, 100-MSPS, 8-channel Analog-to-Digital Converter*, Texas Instruments, 2012. [Online]. Available: www.ti.com/lit/ds/symlink/ads5295.pdf

[20] *LVDS Owner's Manual*, Texas Instruments, 2008, fourth Edition. [Online]. Available: www.ti.com/lit/ml/snla187/snla187.pdf

[21] *Cyclone V Device Datasheet*, Altera, 2015, 2015.12.04. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/cyclone-v/cv_51002.pdf

[22] A. Shoval, D. Johns, and W. Snelgrove, "Comparison of DC offset effects in four LMS adaptive algorithms," *IEEE Transactions on Circuits and Systems*, vol. 42, no. 3, pp. 176–185, Mar. 1995.

[23] *Low Power Five-Channel Digital Isolator*, Silicon Labs, 2015, rev. 1.8. [Online]. Available: https://www.silabs.com/Support%20Documents%2fTechnicalDocs%2fSi865x.pdf

[24] *Digital Isolator Design Guide*, Texas Instruments, 2014, sLLA284A. [Online]. Available: http://www.ti.com/lit/an/slla284a/slla284a.pdf

[25] *ISO722x Dual Channel Digital Isolators*, Texas Instruments, 2015, rev. N. [Online]. Available: http://www.ti.com/lit/ds/symlink/iso7220a.pdf

[26] *Cyclone V Device Handbook Volume 1: Device Interfaces and Integration*, Altera, 2015, 2015.12.21. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/cyclone-v/cv_5v2.pdf

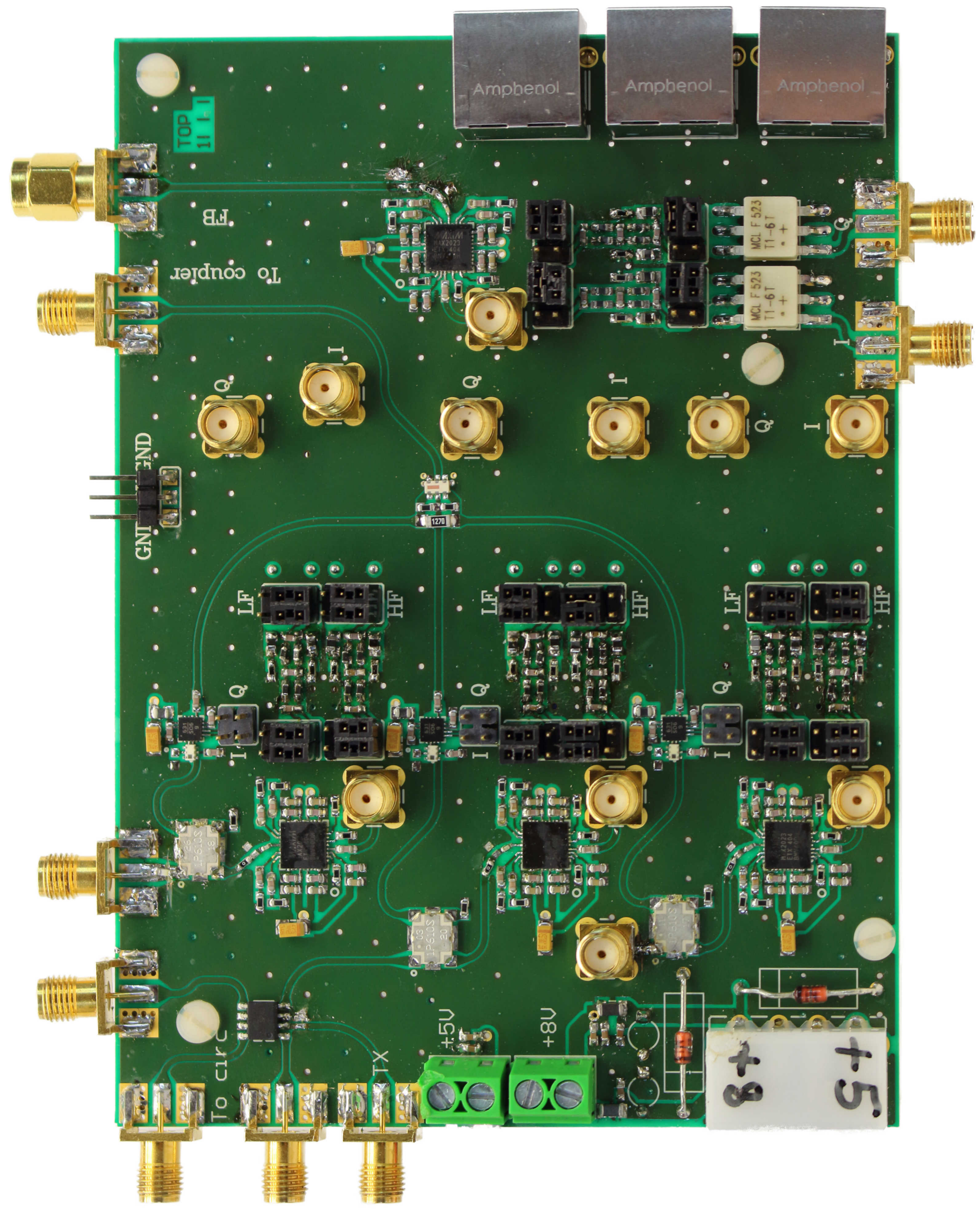
[27] *Altera Phase-Locked Loop (Altera PLL) IP Core User Guide*, Altera, 2015, rev. 1.2. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/ug/altera_pll.pdf

[28] *Octal, 16-bit nanoDAC+ with SPI Interface*, Analog Devices, 2008, rev. B. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD5676.pdf

[29] *Micropower Voltage Reference*, Intersil, 2015, fN7896.3. [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/fn78/fn7896.pdf

[30] *FT232R USB UART IC*, Future Technology Devices International, 2015, version 2.12. [Online]. Available: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

APPENDIX A. UNEDITED CANCELLER PCB



APPENDIX B. THE SPI MASTER VHDL CODE

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
-----------------------------------------------------------------------
ENTITY spi_master IS
PORT(
  clk : in std_logic;
  rst_n : in std_logic;
  control_data : in unsigned(95 downto 0);
  da_value_out : out unsigned(95 downto 0);
  da_value_in : in unsigned(95 downto 0);
  manual_control : in std_logic_vector(2 downto 0);
  SCK : out std_logic;
  SDO : out std_logic := '0';
  SYNC_N : out std_logic := '1';
  LDAC_N : out std_logic := '1';
  RESET_N : out std_logic := '1';
  halt_en : in std_logic := '0';
  SDI : in std_logic;
  accumulator_en : out std_logic := '0');
END spi_master;
-----------------------------------------------------------------------
ARCHITECTURE a1 OF spi_master IS
  type state_type is (send,sample,prepare,channel_switch,halt);
  signal current_state : state_type := sample;
  type sample_array is array (0 to 5) of unsigned(15 downto 0);
  type address_array is array (0 to 5) of std_logic_vector(3 downto 0);

  -- Array to contain 16-bit data values for each channel.
  signal samples : sample_array := (others => (others => '0'));
  -- DAC output channels
  signal addresses : address_array := ("0110", "0111", -- Tap 1 I and Q
                                       "0101", "0100", -- Tap 2 I and Q
                                       "0011", "0010");-- Tap 3 I and Q

  -- Only write to DAC register command is needed
  constant command : std_logic_vector(3 downto 0) := "0001";
  -- Bit vector for the SPI serial data
  -- It will contain the command, address and data.
  signal temp : std_logic_vector(0 to 23) := (others => '0');
  -- Counter to keep track of how many bits have been sent.
  signal bit_count : integer range 0 to 24 := 0;
  -- Counter to keep track of how many channels have been written.
  signal channel_count : integer range 0 to 5 := 0;

BEGIN
  RESET_N <= rst_n;

  -- Keep the clock signal low
  -- when SPI is halted
  SCK <= clk when halt_en = '0' else '0';

  -- Send current DAC values to the custom
  -- instructions handling the nios.
  da_value_out(95 downto 80) <= samples(0);
  da_value_out(79 downto 64) <= samples(1);
  da_value_out(63 downto 48) <= samples(2);
  da_value_out(47 downto 32) <= samples(3);
  da_value_out(31 downto 16) <= samples(4);
  da_value_out(15 downto 0) <= samples(5);

PROCESS(clk,rst_n)
BEGIN
  if rst_n = '0' then
    current_state <= sample;
    LDAC_N <= '1';
    SYNC_N <= '1';
    channel_count <= 0;
    samples <= (others => (others => '0'));
    accumulator_en <= '0';
  elsif rising_edge(clk) then
    accumulator_en <= '0';
    case current_state is

      -- Get new values from the algorithm or from NIOs
      -- depending on the value of manual control bit for
      -- each output pair (two for each vector modulator)
      -- This is the first state in the sequence to update
      -- all channels.
      when sample =>

        -- Input values for Tap 1
        if manual_control(0) = '1' then
          samples(0) <= da_value_in(95 downto 80);
          samples(1) <= da_value_in(79 downto 64);
        else
          samples(0) <= control_data(95 downto 80);
          samples(1) <= control_data(79 downto 64);
        end if;

        -- Input values for Tap 2
        if manual_control(1) = '1' then
          samples(2) <= da_value_in(63 downto 48);
          samples(3) <= da_value_in(47 downto 32);
        else
          samples(2) <= control_data(63 downto 48);
          samples(3) <= control_data(47 downto 32);
        end if;

        -- Input values for Tap 3
        if manual_control(2) = '1' then
          samples(4) <= da_value_in(31 downto 16);
          samples(5) <= da_value_in(15 downto 0);
        else
          samples(4) <= control_data(31 downto 16);
          samples(5) <= control_data(15 downto 0);
        end if;

        -- Set counts to zero. Set LDAC_N low to update
        -- the current DAC register values to the output.
        channel_count <= 0;
        bit_count <= 0;
        LDAC_N <= '0';
        current_state <= prepare;

        -- Halt the state machine
        if halt_en = '1' then
          current_state <= halt;
        end if;

      when halt =>
        if halt_en = '0' then
          -- Continue from start.
          current_state <= sample;
        end if;

      -- Create the bit vector to be sent.
      when prepare =>
        LDAC_N <= '1';
        temp <= (command &
                 addresses(channel_count) &
                 std_logic_vector(samples(channel_count)));
        current_state <= send;

      when channel_switch =>
        channel_count <= channel_count+1;
        bit_count <= 0;
        current_state <= prepare;

      -- Set SYNC_N low so DAC accepts data.
      -- Transfer 24 bits of data on the
      -- following clock cycles.
      when send =>
        SYNC_N <= '0';

        -- Enable signal for accumulator, DAC glitch
        -- has passed by this time.
        if bit_count = 10 and channel_count = 5 then
          accumulator_en <= '1';
        end if;
        if bit_count < 24 then
          SDO <= temp(bit_count);
          bit_count <= bit_count+1;
        else
          -- If all bits have been sent
          -- Raise SYNC_N and switch channels.
          SYNC_N <= '1';
          current_state <= channel_switch;
          if channel_count = 5 then
            -- If all channels have been written
            -- Go get new control value samples.
            current_state <= sample;
          end if;
        end if;
    end case;
  end if;
END PROCESS;
END a1;
```