DIPLOMA THESIS

A FPGA based DAQ system for mobile robot use

Submitted at the Faculty of Electrical Engineering, Vienna University of Technology
in partial fulfillment of the requirements for the degree of
Master of Sciences (Diplom-Ingenieur)

under supervision of

Em.O.Univ.Prof. Dipl.-Ing. Dr.techn. Gottfried Magerl,
Projektass.Dipl.-Ing. Christian Walter Msc.

by

René Paul Hinterberger
Matr.Nr. 0425858
Bruesslgasse 36/23, 1160 Vienna

Date _____

Kurzfassung

Ziel dieser Arbeit war die Entwicklung eines “A FPGA based DAQ system for mobile robot use” (FPGA basierendes Datenerfassungssystem für den Einsatz an mobilen Robotern).
Obwohl dieses Gerät für eine Vielzahl von Anwendungen genutzt werden kann, war das primäre Ziel ein Datenerfassungssystem für Ultraschallsensoren welche auf einem mobilen Roboter befestigt sind. Um die Verwendbarkeit in einem weiten Bereich zu ermöglichen, wurde ein universeller Ansatz bei der Entwicklung des Hard- und Softwarekonzepts gewählt.

Die Hauptanforderungen für eine Verwendung mit Ultraschallsensoren sind: Ein digital zu analog und vier analog zu digital Konverter mit präziser Zeitmessung um Ultraschallsignale senden und empfangen zu können, bei denen Pulskompressionsverfahren für hochauflösende Laufzeitmessungen zum Einsatz kommen. Durch diese hochauflösenden Laufzeitmessungen kann die 3D Position des reflektierenden Objekts berechnet werden. Die Verwendung mehrerer Kanäle erhöht dabei die Zuverlässigkeit des Systems.

Anforderungen an das System sind: ein definierter und stabiler Phasengang, gleichzeitiges Abtasten auf allen Kanälen, geringes Kanalübersprechen, ein hoher Dynamikbereich (10Bit ADC) und hohe Abtastraten (>10MSPS). Um eine maximale Flexibilität des Systems zu erreichen wurde ein FPGA mit „Softcore“ und eine USB Schnittstelle verwendet. Durch den Einsatz des FPGA können rechenintensive Operationen in Hardware abgebildet werden, wodurch der Mikroprozessor entlastet wird. Die USB Schnittstelle ermöglicht eine einfache Verbindung mit einem PC und MATLAB.

Der Prototyp welcher in dieser Arbeit entwickelt wurde, erlaubt es Datenraten von 400MBit/s pro Kanal zu verarbeiten, wodurch sich eine Systembandbreite von 1,6GBit/s ergibt. Durch die Verwendung von pinnkompatiblen Komponenten sind Datenraten bis zu 4,8GBit/s möglich.

Abstract

Goal of this thesis was to develop a working prototype of an “A FPGA based DAQ system for mobile robot use”. Such a device is suitable for a wide variety of applications although our primary target was a data acquisition (DAQ) system for an ultrasonic sensor mounted on a mobile robot. To assure applicability in a larger number of projects a more universal approach was chosen for the design of the hard- and software concept.

The primary requirements for the ultrasonic sensors are: One digital to analog converter and four analog to digital converters, with precise timing to transmit and receive ultrasonic signals. These are further post processed using pulse compression methods for high resolution time-of-flight (ToF) measurements. Using such high resolution time-of-flight measurements the 3D position of a reflection point can be calculated. Additional channels can be used for enhancing the reliability of such a sensor.

Desirable properties of such a DAQ system are: well defined and stable phase, synchronous sampling on all channels, low channel crosstalk, high input dynamic range (10Bit ADC) and a high sample rate (>10MSPS). For maximum application flexibility a FPGA with a softcore and a USB interface was used. Using an FPGA allows implementation of computation expensive processes in hardware offloading the MCU load. The USB interface allows easy connection to a PC running MATLAB.

The prototype developed in this work allows data rates of 400MBit/s per channel resulting in a total system bandwidth of 1,6GBit/s. Using the same software in combination with pin-compatible components data rates up to 4,8GBit/s are feasible.

Table of Contents

Abbreviations

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
A/D	Analog / Digital
BGA	Ball Grid Array
DAC	Digital to Analog Converter
DAQ	Data Acquisition
DDR	Double-Data-Rate
DMA	Direct Memory Access
DSP	Digital Signal Processor
D/A	Digital / Analog
EMCE	Institute of Electrodynamics, Microwave and Circuit Engineering
ESL	Equivalent Series Inductivity
ESR	Equivalent Series Resistance
FPGA	Field Programmable Gate Array
FTDI	Future Technology Devices International Ltd.
GPIO	General Purpose Input/Output
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
ISE	Integrated Software Environment
MPMC	Multi Port Memory Controller
MSG	Message
MSPS	Mega-Samples per Second
NPI	Native Port Interface
PCB	Printed Circuit Board
PHY	Physical Layer
PLB	Processor Local Bus
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
SD	Synchronous Dynamic
SDRAM	Synchronous Dynamic Random Access Memory
THD	Total Harmonic Distortion
THT	Through Hole Technology
ToF	Time of Flight
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VGA	Variable Gain Amplifier
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VCA	Voltage Controlled Amplifier

1. Introduction

Sound waves, which are above the human audibly range, are called ultrasonic waves. The wide range of applications, simplicity, robustness and competitive component pricing of ultrasonic systems often make them the first choice for a lot of applications.

The workgroup “Measurement and Control” at the Institute of Electrodynamics, Microwave and Circuit Engineering, focuses on the application and development of ultrasonic sensors for measurement and control.

Goal of this thesis is to develop “A FPGA based DAQ system for mobile robot use”. It has to be especially tailored for this purpose to allow synchronously triggered sending and receiving of analog signals. This is required for time-of-flight (ToF) measurements. The system has to provide four analog inputs and one analog output with a minimum resolution of 10Bit@10MSPS at each channel. Small size, low power consumption and a USB interface are also required for this application.

A detailed research on existing systems providing the required functionality at an affordable price was the starting point of the development. It turned out that no existing hardware was available to fulfill these requirements. The FPGA evaluation boards for >10MSPS including multiple converters are located in the high performance end of the product spectrum and tend to cost more than one thousand Euros. Furthermore they are usually equipped with high end FPGAs, which are not supported by any free evaluation tools (e.g. Virtex family from XILINX is not supported by ISE-WebPACK[1]).

The following section gives a brief introduction into time discrete systems and sampling theory and may be skipped by the profound reader.

1.1. Theory about time discrete systems

When analog signals need to be processed they are most of the time first converted into a digital signal. The sampling of an analog signal is typically performed at equidistant time spaces $\boldsymbol{\Delta T}$. The sampling frequency $\boldsymbol{fs}$ is the reciprocal value of the sampling time $\boldsymbol{\Delta T}$. The amplitude resolution is defined by $\boldsymbol{n}$, the number of bits available for the digitalization, as well as the full scale range $\boldsymbol{A = 2\text{^}n}$. After digitalization the analog value $\boldsymbol{U}$ of the signal is represented by the closest available digital value $\boldsymbol{V}$.

$$\boldsymbol{U \rightarrow V = m * \frac{A}{n}, m \leq n}$$

The difference between the analog signal value and the corresponding digital signal value is the quantization error. To keep this error low, the resolution should be as high as possible. Contrary to this requirement, high resolution requires complex analog to digital converters which are more expensive.

An important aspect to keep quantization error as low as possible is to adjust the input signal to match the full scale input voltage of the ADC. This pre-processing step ensures the maximum achievable dynamic range at the output of the ADC (see Fig.1 for an example).

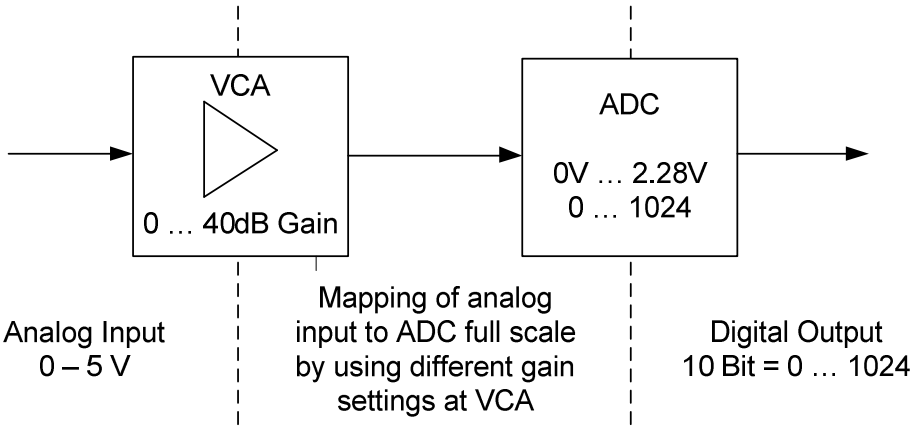


Fig.1: Example of a voltage controlled amplifier to use full scale at small input signals

An example of the quantization can be seen in

Fig.2. If small input signals are forwarded to the ADC without amplification to match the full scale of the input of the ADC, the quantization error is very high. By amplification of these

small signals before forwarding them to the ADC, the quantization error can be reduced drastically.

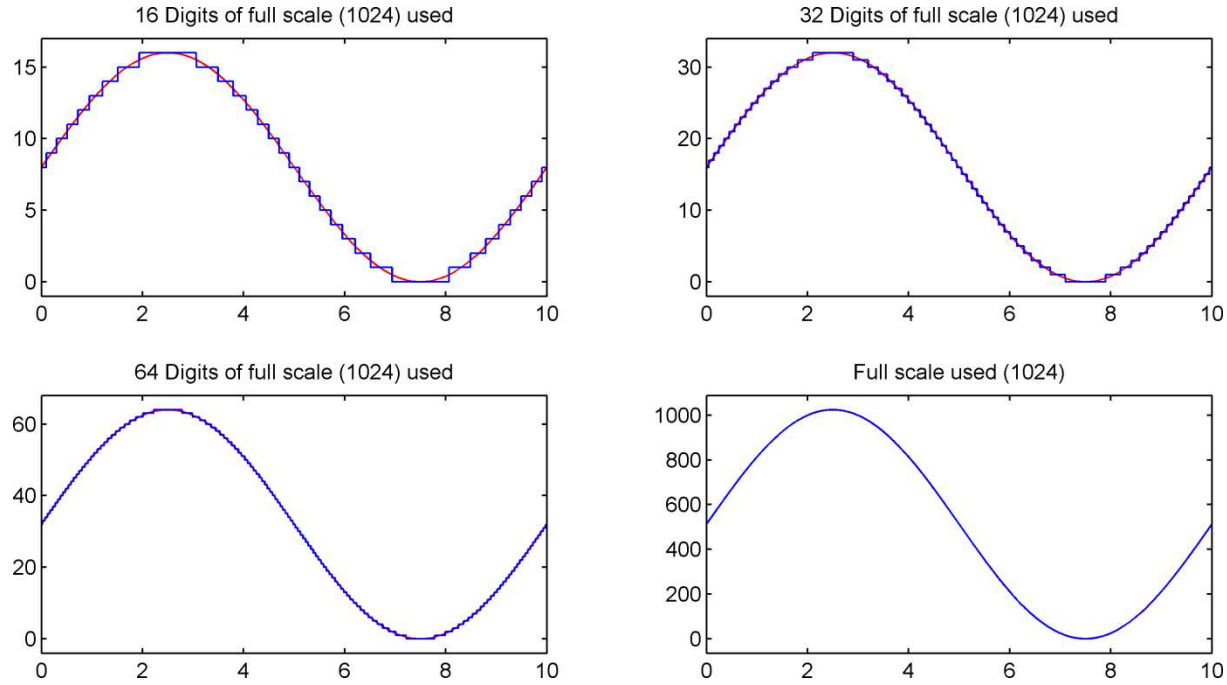


Fig.2: Quantization error at different resolutions

Beside the correct amplification of the input signal, filtering is important to fulfill the sampling theorem of Nyquist- Shannon. If the sampling frequency $\boldsymbol{fs}$ does not match the filter cut off frequency $\boldsymbol{fc} < 2 * \boldsymbol{fs}$, the sampled signal cannot be reconstructed correctly. In case of signal processing the base band signal contains aliased signal components which can impact further processing steps.

Depending on the type of analog to digital converter the resolution and sampling frequency are limited due to technological limitations. Fast conversion is typically done by parallel converters. These generate the corresponding digital value by comparing the input signal with $\boldsymbol{2^n}$ internally generated reference values. As there are also $\boldsymbol{2^n}$ comparators necessary, these converters are used at applications with low amplitude resolution (typical <10Bit). The conversion of the analog input signal is done within one clock cycle and the digital output value is provided directly at the parallel outputs. Therefore these converters are very fast.

Serial analog to digital converters require less than $\boldsymbol{2^n}$ comparators and usually have lower prices. There are several different architectures possible like the Single-Slope-Converter, the

Sigma-Delta-Converter or the Successive approximation ADC. The serial converters process the analog input signal in several steps to generate the corresponding digital output value. They are typically much slower in conversion compared to the parallel ones.
Analog to digital converters obtain the digital signal by using a “sample and hold process”. This means the analog input is sampled at the clock edge and is held for digitalization until the next clock edge arises. Therefore the clock edge directly defines the moment when the analog signal is sampled. Looking at stationary input signals, a variation of the sampling point does not influence the conversion result. At transient signals a variation in the clock signal leads to an error in the digital signal. In this case the sampling at different time point’s results in different digital signal values. A clock jitter therefore can lead to unwanted distortions in the digital output signal and needs to be kept as low as possible.

When looking at several ADCs working next to each other (like in the configuration used for this thesis) another error can occur. Unwanted coupling of the input signal from one channel to other channels generates errors at the output of the ADCs. There are several possible ways for the input signal to couple from one channel to others:

- coupling at the common power supply
- ground loops
- capacitive and inductive coupling in the layout and integrated circuits

The crosstalk attenuation is the “isolation” between the inputs of the system. As the coupling effects are frequency dependent and often appear in combination, the crosstalk attenuation needs to be examined at the full frequency range which the circuit was designed for.

1.2. Hardware / Software Codesign

The hardware / software codesign is described by [2] as a synonym for the development of the digital part of a complex system in software and hardware. During the concept phase it is necessary to define which parts need to be realized in software and which in hardware.

The “co” of codesign can be interpreted as “together” or as “coordinated”. To find the optimal split between software and hardware, several different design approaches have to be taken into consideration. An assessment of each design approach leads to the most practical realization for the system under the given circumstances.

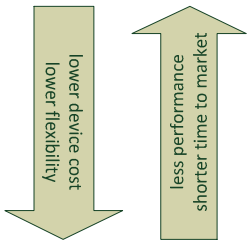
Due to continuous progress in technology of integrated circuits, it is possible to implement more complex tasks into integrated systems and therefore gain performance and lower cost simultaneously.

Complex functions can be realized by using a variety of components that differ in their characteristics regarding:

- time to market
- flexibility (capable to handle different applications)
- device cost
- performance

Components which can be used to realize the required functions are:

- RISC processors
- DSPs
- FPGAs
- ASICs



RISC processors do have much higher flexibility compared to DSPs or FPGAs.
Short development time and low device cost at low quantities make it first choice for a lot of applications. They are typically designed for controlling of processes and do have comparatively low data throughput.

When looking at performance and data throughput, an ASIC is the most efficient way to realize a defined function. It is optimized to fulfill the required tasks and has no unnecessary functions which are not used. The device costs are lowest at high volumes. The disadvantages are the necessity for big investments during the development and a long time to market as the silicon is a custom design. Once an ASIC is designed it can only be used to realize the defined functions. A change in the required functionality leads to a redesign of the ASIC.

Performance and flexibility cannot be maximized at the same time and form the trade-off at the realization. Other criteria as time-to-market and power consumption also need to be taken into consideration as they can exclude some type of ICs at some applications.

1.3. Theory about FPGA & VHDL

A **F**ield **P**rogrammable **G**ate **A**rray is an integrated circuit consisting of matrix organized logic blocks with horizontal and vertical connection structures in between. It has programmable interconnect points for the required connections between the blocks. At the edges of this matrix so called I/O blocks are placed to connect the logical cells to the periphery (see Fig.3).

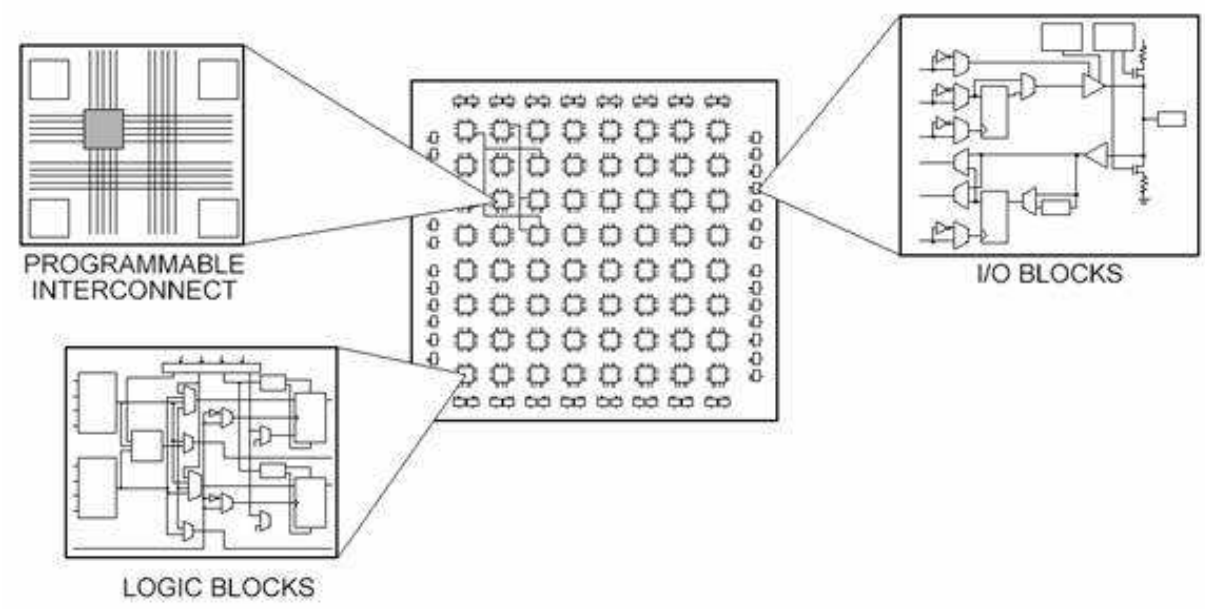


Fig.3: Basic setup of an FPGA [3]

Basically any logical function can be generated by combining simple NAND gates. But the number of necessary NAND gates rises drastically with increasing complexity of the logical function. The logical blocks inside an FPGA are much more complex than simple NAND gates. They allow realization of simple and also complex functions with a reasonable number of logical blocks in combination. As the logical blocks can be combined in any order, it is possible to generate several functions completely independently from each other in one single FPGA chip. They are acting like several individual chips inside one package.

Without any program inside, an FPGA does not have any function at all. It just consists of unconnected logical blocks. During programming of the FPGA the interconnect points are set. Furthermore the configuration of the I/O blocks and logical blocks is done to realize the desired function. The huge amount of logical blocks in a state of the art FPGA makes it

possible to realize very complex functions, e.g. a complete microcontroller inside an FPGA. Even several independent microcontrollers can be realized inside one single FPGA.

The I/O blocks are also programmable and beside the configuration as input and output, they can be configured to work with different pull up/down resistors, voltage levels and impedances. The program containing the configuration is a binary file generated by a compiler. Depending on the manufacturer and type of FPGA, there are several different compilers available on the market. Nevertheless there is a manageable amount of programming languages for FPGAs (input for the compiler) which are popular. One of the most common programming languages, which is used for programming FPGAs, beside some other applications, is VHDL.

VHDL is the acronym for **V**ery **H**igh Speed Circuit Hardware **D**escription **L**anguage. It is a hardware description language which allows describing the behavior of digital systems in text form. VHDL is defined since 1987 in the IEEE 1076 standard [4].

2. Concept

When designing the “FPGA based DAQ system for mobile robot use” with respect to the requirements which were given, a top down design was started with a black box (see Fig.4).

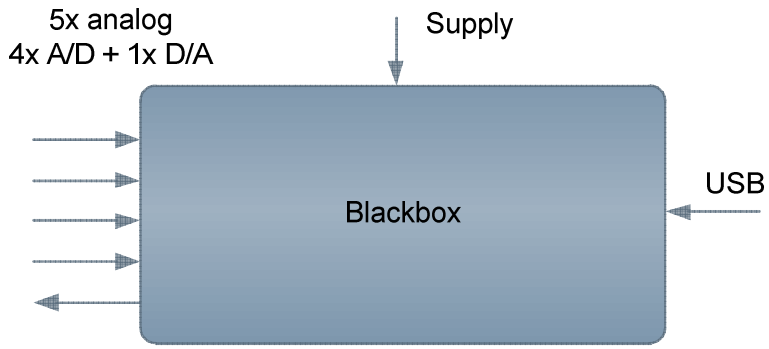


Fig.4: Black box approach with defined interfaces

The first idea was to use a microcontroller in combination with four A/D, one D/A converter, a fast RAM and an FPGA to perform the required signal processing. As the data rates at the required sampling frequency and resolution get very high when recording four channels in parallel (400MBit/s), a realization by using a microcontroller for handling the data seemed to be not feasible. Furthermore the RAM had to be either dual ported or all data had to be forwarded to the FPGA by the microcontroller (see Fig.5).

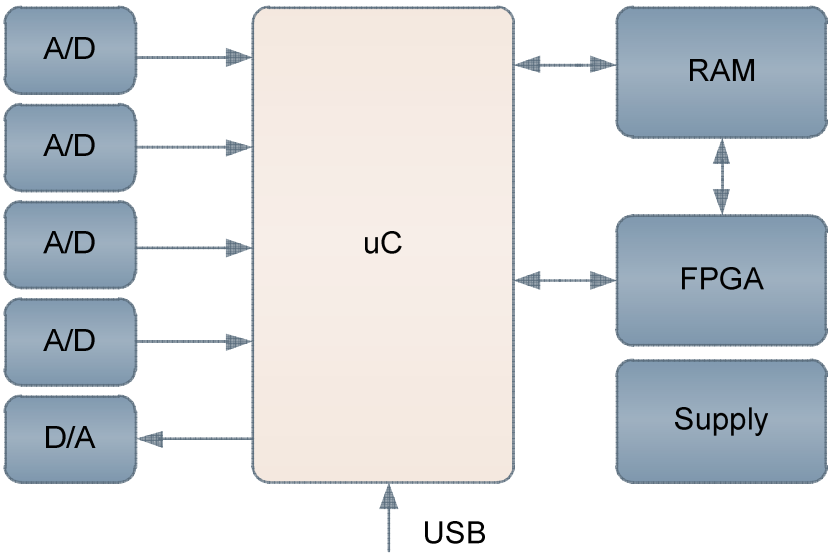


Fig.5: Block diagram of the setup with uC handling all data

Although state of the art microcontrollers provide DMA in combination with external memory, a better approach seemed to be using an integration of the microcontroller into the FPGA – a so called “softcore”. In this case the microcontroller can be bypassed for the data transfer during recording and is not occupied with the handling of the data during recording. The data can furthermore directly be pre-processed by any functions implemented in the FPGA before storing it to the memory. At this approach a closer look to the correct Hardware/software codesign was the next logical step.

2.1. Hardware/Software Codesign

The hardware/software codesign defines which parts of the necessary function are handled by hardware and which in software. Using an FPGA allows high flexibility regarding the implementation of some logical functions into the FPGA instead of placing them on the PCB as real physical components. As an FPGA was part of the concept from the beginning for data processing, it was obvious to implement further functions inside the FPGA in order to reduce the number of external components. Some of the advantages and disadvantages of using an FPGA for this application can be seen in Tab. 1.

Advantage	Disadvantage	Comment
Performance		Sampling rates >> 10MSPS can be achieved
Flexibility		“Hardware” can be easily changed by reprogramming the FPGA
Custom IP		User defined functions like clock-, memory and address generation or complex signal processing can easily be implemented
Pre-defined IP’s		Customizable “softcores” and memory interface including DDR refresh pre-defined available
	High power consumption / several different voltages required	High efficient switching power supply necessary to keep total power consumption low
(Cost)	Cost	Basic FPGA necessary for signal processing. But additional cost to increase FPGA size to implement additional functions, are lower than external components required for this functions
	BGA package at “mid size” FPGA	Use of evaluation boards including minimal functionality and workable pin out required
	Complex development tools	Reduce project setup to a minimum inside the development tools, to keep effort manageable

Tab. 1: Advantages and Disadvantages of an FPGA for this application

The analog input signal pre-processing as well as the A/D converter need to be realized as external components, but when it comes to controlling these external components and processing the input data, many functions can be implemented in form of programmable HW inside the FPGA.

The development tools from Xilinx provide some predefined softcores acting as a customizable complete microcontroller. Implementing such a softcore can avoid an additional external microcontroller and gives very high flexibility, as the HW of this microcontroller inside the FPGA can easily be tailored to the required function. Although a microcontroller would not be necessary for this application, as the complete functions could be implemented in VHDL as well, it is much more convenient to use one programmed in C (compilers are available inside the Xilinx development tools). Other IPs provided by Xilinx can be used for interfacing the external memory (see a block diagram of this setup in Fig.6).

Last but not least a custom generated IP programmed in VHDL can take care of the data processing and handling of the data transfer from the ADC/DAC interface to the memory interface to avoid high loads at the microcontroller.

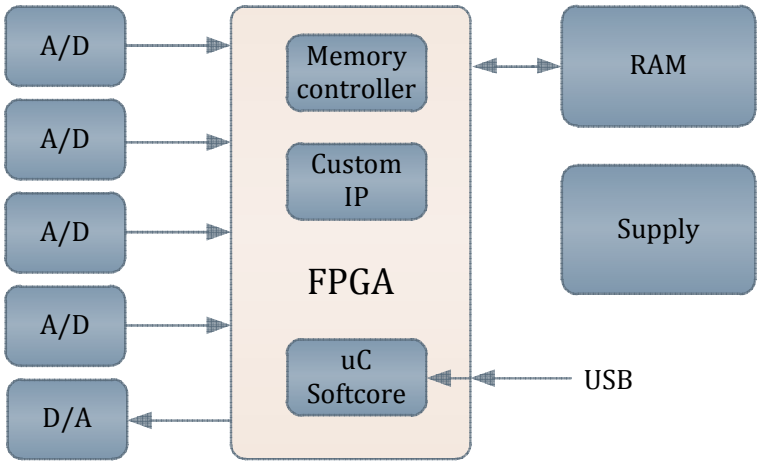


Fig.6: Block diagram of setup with uC inside FPGA,
all data handled by memory controller and custom IP

To optimize the function blocks inside the FPGA a suitable partitioning of these blocks has to be done. This partitioning will be discussed in the next chapter.

2.2. Partitioning

At the design of complex digital circuits the process of clustering objects into groups in a way that the object given function is optimized with respect to a set of design constraints is called partitioning.

During concept phase the functions which had to be handled inside the FPGA to reduce external components and increase performance and flexibility, were defined. A list of these functions can be seen in Tab. 2(Functions implemented as custom IP are marked in green).

Function	Description
Memory interface	Physical interface to the installed memory
PC Interface	Interface to the external USB driver chip
PLL	Reference clock generation (single external crystal clock source)
Clock generator for A/D & D/A conversion	Generation of user selectable clock rates for external A/D and D/A converters (low jitter required for precise data acquisition)
Softcore	Microcontroller to control the data acquisition and communicate with the PC
Timer	Peripheral device for the softcore for time dependent tasks
Reset	Ensure correct startup of IPs inside the FPGA
Cache	Buffer of data from and to the DDR memory (buffer refresh cycles of DDR)
Address generator	Required for complete decoupling of softcore from memory during fast data acquisition
Data interface	Mapping of 40 bit input data to the 16 bit memory interface (possibility for data pre- processing)
GPIO	Providing digital inputs and outputs (e.g. to configure external assembled integrated circuits)

Tab. 2: List of functions implemented in the FPGA

Fig.7 shows a block diagram of the function blocks inside the FPGA. The blocks marked in red are responsible for handling the data during conversion. These blocks need to be fast and capable of processing high data rates. Except the memory interface, all red blocks are realized as one custom IP having one dedicated data and address bus for handling high data rates independently from the other blocks.

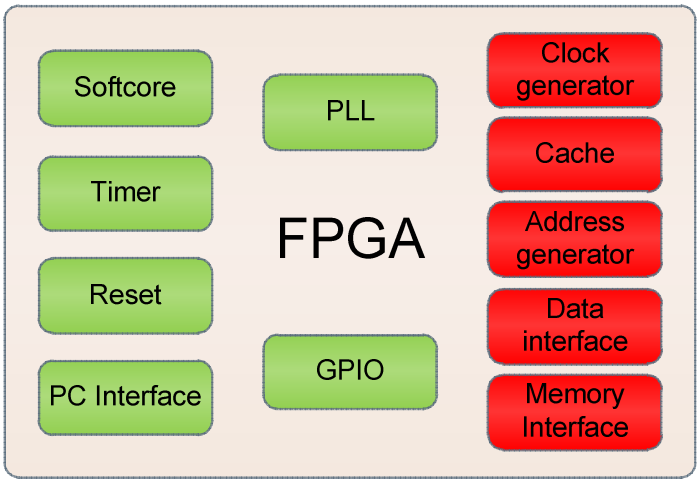


Fig.7: Block diagram of functions implemented in the FPGA.
Blocks marked in red are involved during high speed data recording.

2.3. Hardware Concept

The HW can basically be split into six main sub elements:

- FPGA
- Power Supply
- PC Interface
- ADC
- Analog interface and VGA
- DAC

For each of these elements an investigation based on the requirements was made to evaluate the components that suit best for a universal design concept including an ultrasonic signal processing application.

2.3.1. FPGA

To develop a complete hardware including a layout for an FPGA and RAM is a very time consuming task as most of the FPGAs are available only in a BGA package. This makes the layout very complex and fast prototyping almost impossible. Not only the package is critical to handle, but also certain layout rules to keep the required track impedances and lengths required by state of the art DDR memories need to be followed with great caution. To shorten this process it was decided to use an “off the shelf” FPGA evaluation board to avoid the complex layout and assembly of the FPGA and memory chip.

As there is a very high variation of these boards available on the market some criteria had to be defined to sort out the suitable boards. A list of these criteria can be found in Tab. 3.

Criteria	**Reason**
small size	Suitable for mobile application
Interfacing connectors on main board manageable for hand soldering	Benefit of easy manufacturing is lost otherwise
min 30MB RAM	Capability of recording an ultrasonic signal at 4 channels with 10Bit@10MSPS for ~ 500ms
Fast RAM interface	Storage of unprocessed data at 400MBit/s
Affordable price	No need to argue

Tab. 3: List of criteria for selection of the FPGA evaluation board

The two market leaders for FPGAs are Xilinx [5] and Altera [6]. One of the advantages from Xilinx is the availability of a big IP catalogue including several different memory controllers. The Spartan 6 family includes also an integrated memory controller block to provide optimal memory performance without allocating a big amount of logical cells for this function. Therefore Xilinx Spartan 6 was preferred at the research of evaluation boards. One of the most promising boards for this purpose seemed to be a board from ZTEX [7]. They are available in several FPGA and RAM sizes. It is also possible to generate a complete stand alone device as some of their boards do have an SD card connector to store the FPGA configuration. An external microcontroller with EEPROM is also onboard to install the boot loader. For the development of the prototype finally the ZTEX board “1.11c” was chosen [8]. The FPGA assembled on this board is a XILINX Spartan 6 XC6SLX25 in combination with a 64MB DDR SD RAM which can handle data rates up to 800MByte/s. More powerful but still pin compatible versions of the board are also available at some higher costs. They are equipped for example with up to 128MB DDR2 RAM and XILINX Spartan 6 XC6SLX150 which has six times more logical cells than the XC6SLX25. During the software development it was confirmed that the XC6SLX25 is big enough to handle all required functions.

2.3.2. Power supply

Although there is a power supply unit available from ZTEX which matches to the USB-FPGA-Module, it was decided to attach a new design for the switching power supply. The reason for this was the concern about interference of the ZTEX power supply with the front end amplifier. The ultrasonic signals which will be received by the microphones and amplified with the front end are in the range of 50-100 kHz. The ZTEX power supply uses fixed off-time, current-mode-controlled buck switching regulators which may generate distortions in a similar frequency range as there is no fixed switching frequency defined. Furthermore the design consists of three independent regulators which are not synchronized. Therefore the generated ripple of each single converter has an undefined phase shift to the others and can therefore generate unpredictable frequency components.

Another reason is the power supply of the components which are not placed on the ZTEX board need to be provided as well (e.g. A/D and D/A converters). To reduce possible distortions and the required layout space, a triple output step-down switching regulator with 550 kHz fixed switching frequency from Allegro was chosen (A4490).

2.3.3. PC Interface

The interface to a PC is required to enable down- and upload of data to and from the memory, and to send commands from and to the electronic.

The control interface does not need to be fast as only simple commands are sent from the PC to the electronic and back. When looking at the down and upload of data, speed does matter as the required amount of data is in a range of several megabytes (the chosen FPGA board from ZTEX does have 64MB RAM which can be increased to higher values at different board versions).Therefore a dedicated USB 1.0 interface was added to the design by using an FT232RL from FTDI (see [9]). It is used as standard serial interface with a data rate of 921,6kBit/s. It can easily be controlled at the PC (virtual com port) and also at the FPGA (standard UART at the softcore).

For higher data transfer rates, the onboard USB 2.0 interface of the ZTEX board can be used (up to 480MBit/s). To do so there is no change in the hardware necessary, but some modifications in the software on the PC (USB 2.0 driver) and in the FPGA binaries.

2.3.4. Analog to Digital Converter

Due to the high number of IO's of the FPGA board, the interface of the A/D and D/A converter can be a parallel interface. This increases the number of tracks but nevertheless reduces layout complexity. Most of the FPGA (also Spartan 6 which is used here) do have high speed serial interfaces onboard which can handle data rates up to several GBit/s. Nevertheless when it comes to designing a layout for these interfaces, the design gets very tricky due to the impedance requirements of these tracks. A parallel interface reduces the clock frequencies dramatically and simplifies the layout as no specific impedances are necessary for the data lines. The clock line still needs to be routed carefully to avoid reflections which can lead to multiple clock triggers at low rise times.

The chosen analog to digital converter for this application is a MAX1183. It is designed for high-resolution imaging, multichannel IF sampling and also ultrasound application. The resolution provided is 10Bit at a conversion speed of 40Msps. There are several pin-compatible versions available. The speed grade can be increased by simply replacing the chip (up to 120Msps is possible by using MAX1190).
Further advantages for this device are: low power consumption, 59.6dB SNR at f_{IN} = 20MHz, a compact design (two converters in one 48-pin TQFP package) and 0.02dB Gain and 0.25° Phase matching of the converters at 20MHz. (see also datasheet [10])

2.3.5. Analog interface and Variable Gain Amplifier

To keep the distortions as low as possible the input of the analog to digital converter is differential. All disturbances which are picked up at the differential signal lines are attached to both lines equally and therefore do not affect the signal as long as the maximum input level of the next stage is not exceeded. The input signal coming from the connected microphones is single ended and the signal level does not match to the dynamic range of the ADC. Therefore an amplifier is necessary to match the input signal to the converter. As the signal level provided by the microphones may vary depending on the acoustic attenuation of the sent signal, a variable gain amplifier makes it possible to match the received signal level to the full scale input of the ADC. The additional equivalent noise caused by the quantization error of the ADC can therefore be reduced and the total performance of the system during the measurement increased. Furthermore when looking at other applications a higher dynamic range is more feasible.

In this application a variable gain amplifier with integrated input buffer from Burr-Brown (Texas Instruments) was chosen to provide the necessary amplification and input matching (single ended input – differential output).

At the input and output of the amplifier appropriate filters were placed to reduce the influence of low frequency distortion (below 20kHz) as well as anti aliasing effects at the ADC. These hardware filters may need to be adapted when using the system for other applications.

2.3.6. Digital to Analog Converter

The digital to analog converter is necessary to generate an output signal which can be sent in the final application as “ping”. As the waveform of this signal shall be “free programmable”, a real digital to analog converter which gets its data from the memory seemed to be most suitable. As there are no special requirements to this converter a 10 Bit type with 20Msps and parallel interface was chosen (AD5433 from Analog Devices [10]).
Also in this case the chosen type has the benefit to provide pin-compatible versions with higher bandwidth and data rates. To decouple the output pin from the DAC and provide a low impedance board output with some ESD protection, a simple rail to rail amplifier was added at the output.

2.4. Software Concept

There are three different types of software necessary to implement the requested functionality (three different programming languages):

- The user interface on the PC shall be provided in **MATLAB**
- The configuration and communication with the PC is done by using a softcore microprocessor inside the FPGA programmed in **C** (necessary compilers are provided by the Xilinx development environment)
- The custom IP which handles the data transfer, cache, address mapping and clock generation is “programmed” in **VHDL**

The main functions supported by the interface between MATLAB and the softcore were defined during the concept phase of the development. Every main function can be started by a single command and is proceeding according a defined protocol afterwards. An overview of the commands supported by the softcore can be found in Tab. 4.

Main function	Description
Power on	Switch on power supply for preamplifier, external 2.5V and 5V supply, enable VCA, enable ADC and DAC
Standby	Switch off supplies and disable VCA, ADC and DAC
Config	Enter configuration mode: In this mode the softcore expects the configuration information about requested sampling rates, number of samples to be transmit and received and the digital gain settings of the preamplifier
Download	Start download procedure (transfer data from PC to onboard memory)
Upload	Start upload procedure (transfer data from onboard memory to PC)
Measure	Start measurement (output and input of DAC/ADC data at the configured sampling rate and number of samples)
Reset	Reset system

Tab. 4: List of main functions supported by the softcore

The interface between the softcore and the custom IP is more simple but was defined at a later stage in the development because the design of the IP, as well as the memory interface had to be changed several times to achieve the required functionality. Nevertheless the configuration shall be mentioned here as follows:

It was realized with two HW signals as well as ten 32Bit registers. The first HW signal is connected to a GPIO pin on the softcore (configured as output) and also to a physical IO port pin of the FPGA to trigger the measurement. The second HW signal is connected to an interrupt pin at the softcore to provide an interrupt signal as soon as the measurement is finished. Four of the ten 32Bit registers are used to set the start and stop address of the transmit and receive data signals in the memory. The other six registers are used for the configuration of the clock speed for the A/D and D/A converters, as well as diagnosis and feedback during SW development.

3. Details on the Hardware Design

The main PCB has two layers with SMD components placed on both sides. The placement of the integrated circuits and signal paths on one side and only minor tracks and a ground plane on the other side result in a very robust design. The power supply filtering capacitors were placed as close as possible to the integrated circuits on both sides of the PCB. The FPGA evaluation board from ZTEX can be placed piggyback on the main PCB by THT connectors. The sensitive analog signal processing consists of the filter and amplifier for each channel and had to be placed at some distance to the switching power supply to avoid coupling of distortion signals. The digital to analog converter and the USB interface are less distortion sensitive and do not have placing restrictions. Beside the switching power supply, a linear voltage regulator provides 5V and 2.5V to supply the preamplifier and if necessary also the MEMS microphones with a low noise supply voltage. Fig.8 shows a general block diagram of the main PCB with the ZTEX FPGA Module.

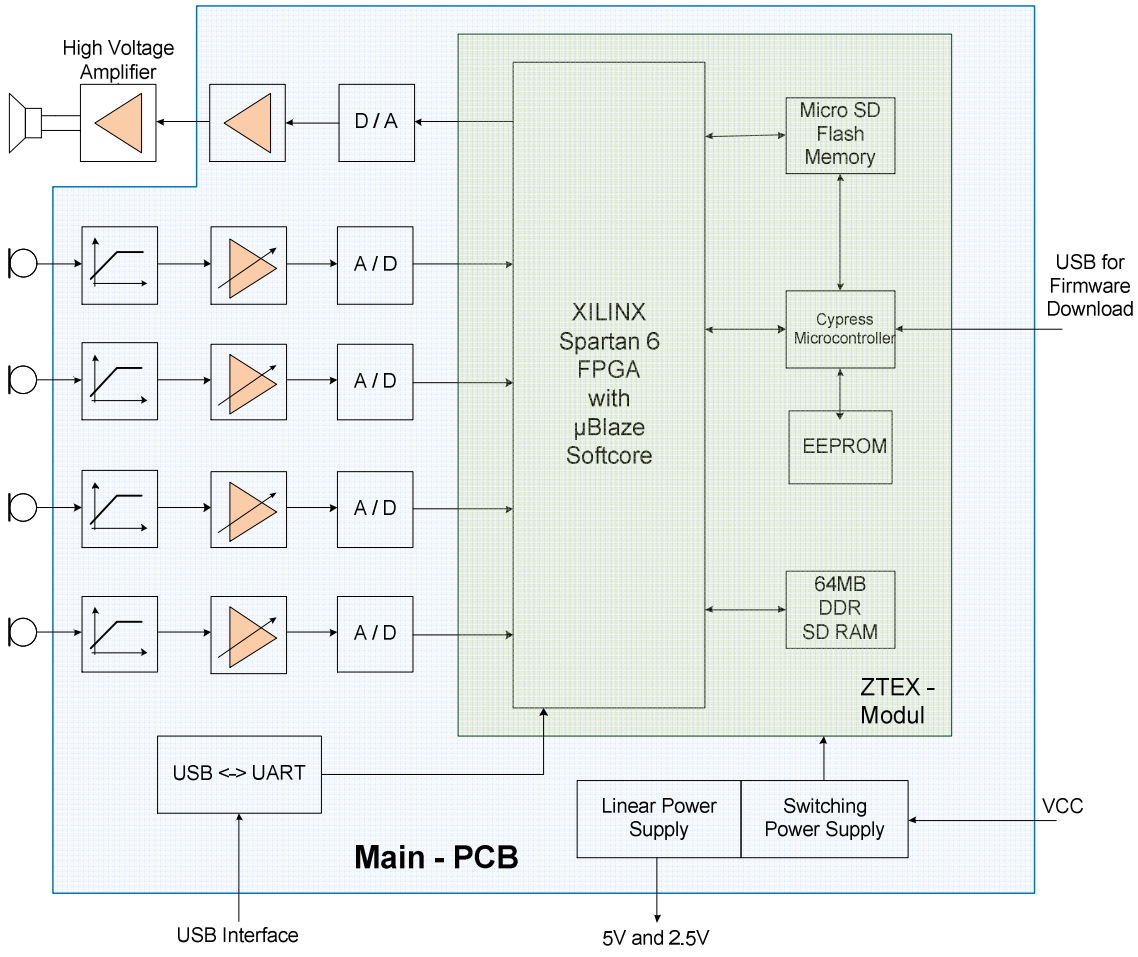


Fig.8: Block diagram of hardware (complete setup)

A more detailed block diagram of the functions included on the main PCB can be found in Fig.9. The control lines of the ADC and DAC as well as the data lines are all connected to the FPGA module and are routed inside the FPGA to the softcore.

Beside the interfaces to the converters, amplifiers and USB driver some additional interfaces for trigger, power signal and debug are required. The trigger line is an output signal from the softcore which indicates the start of a measurement. Once an ultrasonic sound signal is transmitted, it is continuously attenuated during propagation. To enable some compensation of this attenuation a simple capacitor charging circuit provides a time dependent voltage (exponential function) which can be used to increase the gain of the voltage controlled amplifier during the measurement. The input voltage for the VCA can also be set to a stationary value which is adjusted by a simple potentiometer on the PCB.

When using the device for ultrasonic applications at the EMCE an external high voltage amplifier has to be used to drive the ultrasonic transmitter.

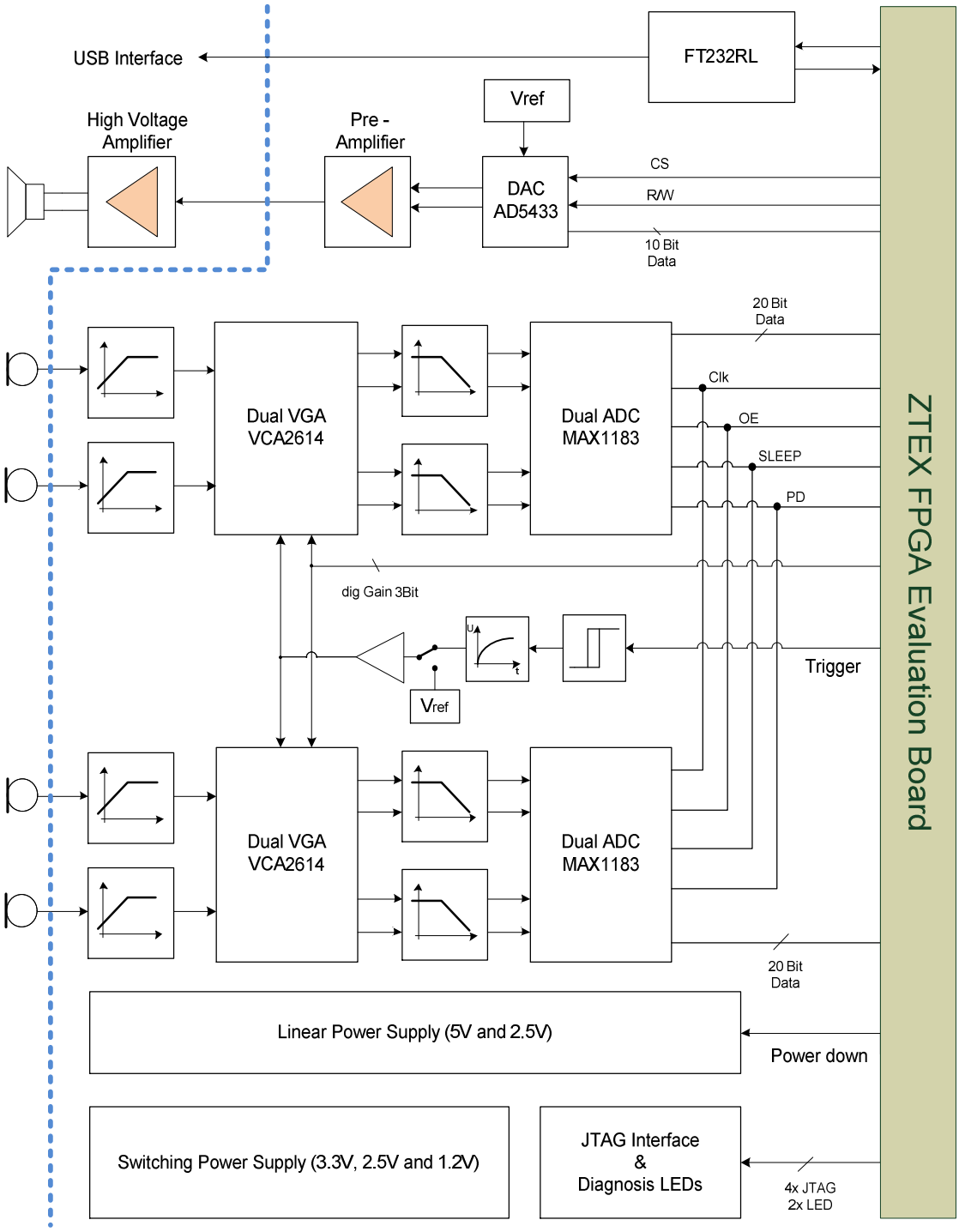


Fig.9: Detailed block diagram of hardware

3.1. FPGA Board

As mentioned in the concept the “ZTEX USB-FPGA-Module 1.11c” [8] was used for the design. See Fig.10 for a block diagram and Fig.11for a picture of the real device.
Further details can be found on the ZTEX website [7].

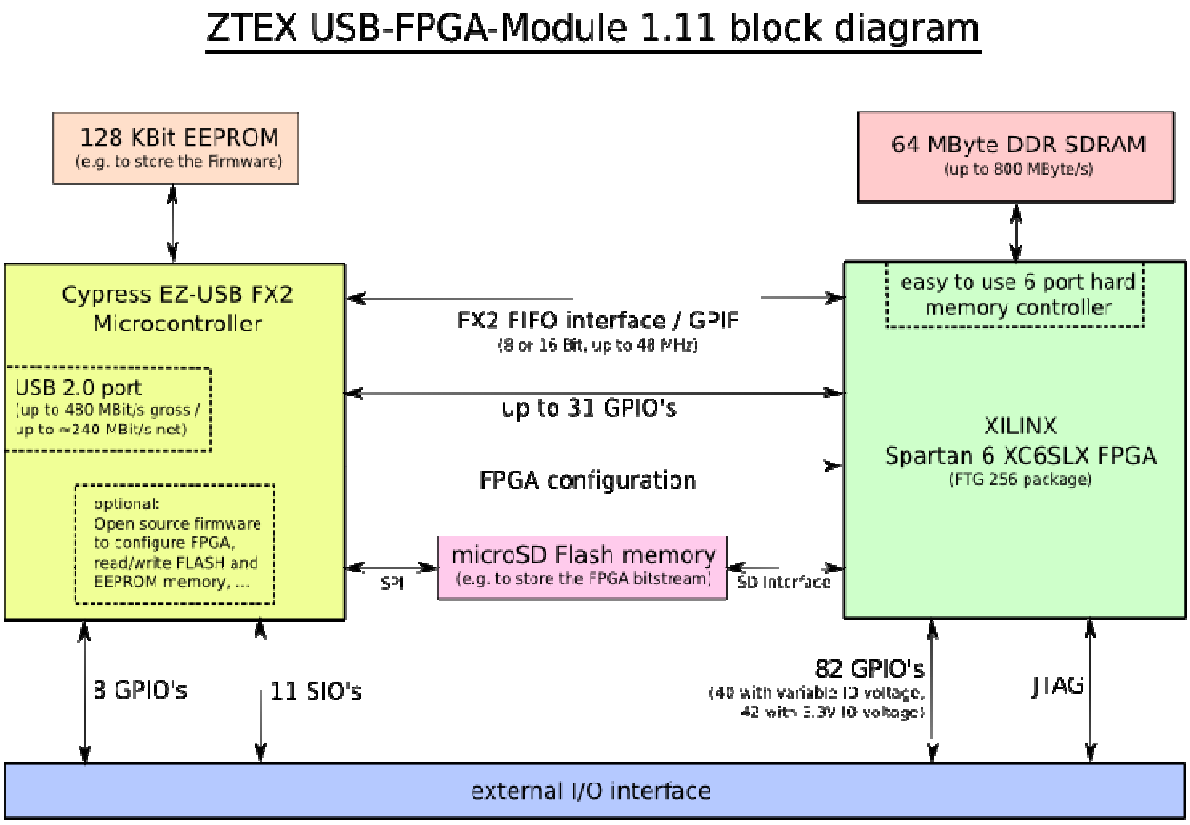


Fig.10: ZTEX USB-FPGA-Module 1.11 block diagram [8]

Fig.11: Picture of ZTEX USB-FPGA-Module 1.11c [8]

The FPGA which is assembled on the Xilinx USB-FPGA-Module 1.11c is a Spartan 6 “XC6SLX25” containing 24,051 Logic Cells [11]. Detailed data can be found at [5]. Directly attached to the Spartan 6 is a 64MB DDR SD RAM from Micron which can handle data rates up to 800MByte/s. The Cypress microcontroller, as well as the flash and EEPROM memory are necessary to enable stand alone operation. During power up the microcontroller starts with a boot loader which is stored in the EEPROM. The “bitstream” for the FPGA is then loaded from the micro SD flash card into the FPGA. The FPGA can also be programmed by a JTAG interface which is connected to a JTAG connector on the main PCB. This interface was also used during development as it is very convenient for debugging. It was disabled after finalization of the software (prototype operates in standalone mode). For measurements at mobile robot application the device had to be able to run in a standalone mode, using a single connection to the PC only for controlling the device.

The onboard USB 2.0 interface is used for downloading the boot loader to the EEPROM and the “bitstream” to the micro SD Flash memory. It is also possible to extend the software of the Cypress microcontroller and use this USB 2.0 interface for communication with the FPGA at high speed rates (up to 480MBit/s). As the software effort increases drastically in this case also on the PC to handle the USB high speed interface, this option was not used for the prototype.

3.2. Power Supply

The power supply of the device contains a linear and a switching power supply. As the board should have low power consumption, most of the supply voltages are provided by the

switching voltage regulator. The linear voltage regulator is only used for supplying the analog frontend as well as the external active microphones (in case of ultrasonic application). As these parts only need to be supplied during a measurement the disadvantage of a very low efficiency at high input voltages can be accepted. The big advantage of a low distortion power supply at these parts of the circuit prevails. The linear regulation is done in a first step to 5.0V by using a L4941BDT low drop voltage regulator (450mV drop voltage @ 1A typ.) followed by a TPS73025 2.5V regulator. The 2.5V voltage is enabled by a control signal from the softcore inside the FPGA. The 5V remains on all time as the analog frontend amplifier supports a power down mode with low current consumption (<2mA per channel). This linear voltage regulator together with the used polarity protection diode defines the minimum and maximum supply voltage of the electronic (V_{CCmin}= 6.1V, V_{CCmax}= 30V).

The main current is provided by the switching power supply. It is used to supply all other parts including the FPGA, except the analog pre amplifier. The IC used for this circuit is the A4490 from Allegro (capable to handle input voltages from 4.5V to 35V).The converter topology is a classic buck converter with external passive freewheeling diodes but internal power FETs. Due to its 4x4mm QFN package it is very compact in size. It provides three independent outputs which are clocked by a fixed frequency of 550 kHz at different pulse widths, to individually control the output voltages. The three clock signals are shifted 120° in phase to reduce the current ripple and distortions on the power input. The maximum output current is internally limited to 2.0A per channel.

Appropriate voltage dividers are used to generate the required voltages with respect of the regulator feedback voltage of 0.8V:

$$\text{4k7 \& 1k5: } \frac{0.8V*(4700+1500)}{1500} = 3.307V$$
$$\text{4k7 \& 2k2: } \frac{0.8V*(4700+2200)}{2200} = 2.509V$$
$$\text{500R \& 1k: } \frac{0.8V*(500+1000)}{1000} = 1.2V$$

As switching power supplies always include high current paths, ripple and distortion signals are highly dependent on the used components and the correct layout. To minimize these effects the coils, freewheeling diodes and filter capacitors were selected carefully to match for this application.

Coils: “IHPL-2525” series from Vishay -> high current, shielded setup and lowest DCR/µH in this package.

Freewheeling diodes: “DFLS230L” from Diodes Incorporated -> “Schottky Barrier” type with high current capability and low forward voltage drop.

Filter capacitors: “GRM Series” from Murata -> Low ESR and ESL

The layout recommendations in the datasheet of the A4490 were tried to be considered to minimize the current flow path during on and off phase of the internal FET [12].

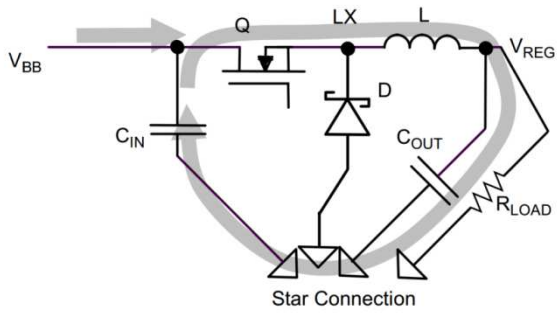


Fig.12: Current flow path at FET on [12]

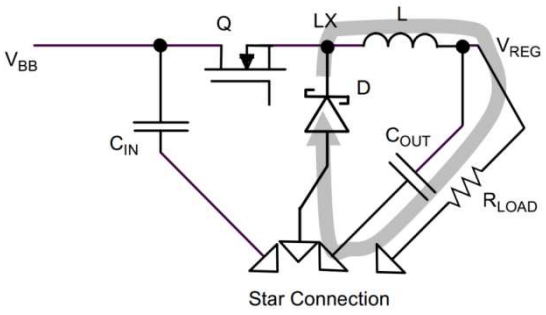


Fig.13: Current flow path at FET off [12]

It was possible to reduce the size of the complete switching power supply layout below 25x20mm by using both side assembly (see Fig.14 for a picture of the switching power supply layout area in ~ 1:1 scale).

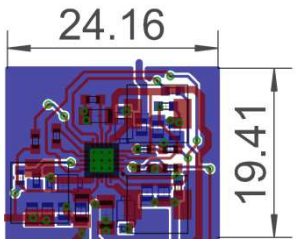


Fig.14: Layout of switching power supply

3.3. PC Interface

The PC interface was kept as simple as possible as there is an existing high speed USB interface available on the FPGA evaluation board from ZTEX. A simple UART to USB chip from FTDI [9] seemed to be the most convenient solution to enable a simple interface to the PC and the electronic. The used FT232RL allows data rates up to 1MBit/s on the UART interface of the electronic (available as part of the softcore), and a virtual com port on the PC side (drivers for Windows provided by FTDI).

3.4. Digital to Analog Converter

The DAC used in the setup (AD5433) is a typical R-2R digital to analog converter with a parallel interface and a resulting current output. In Fig.15 [10] the setup of the R-2R network inside the DAC can be seen.

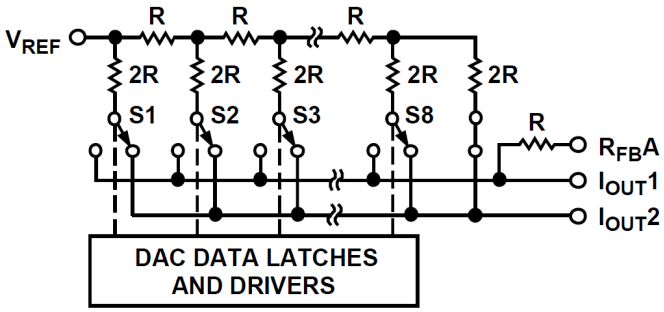


Fig.15: R-2R network inside the digital to analog converter [10]

The output of the DAC in the standard configuration is a current signal. To get a corresponding voltage signal, a voltage switching mode has to be used. By using this configuration a positive reference voltage results in a positive output voltage. Therefore a single supply operation is possible. The output voltage of this configuration is directly dependent on the connected impedance. To avoid this dependency and to provide an independent low impedance output, an additional buffer is necessary. This buffer can also be used to amplify the output signal. As the input is directly connected with the resistor network, the impedance of the input is dependent on the code seen by the R-2R network. Therefore the input has to be driven by a low impedance source to provide correct corresponding output voltages at different codes. At the prototype the input is driven directly by the supply voltage (+5V) buffered by a low impedance L/C filter.

A rail-to-rail, single supply, wideband operational amplifier (AD8601) is used to buffer the output of the DAC and amplify by a factor two. The maximum dynamic range of the output

in this DAC configuration is only 0…V_{IN}/2 if no amplification is used. In Fig.16 the connection of the impedance buffer as mentioned in the datasheet [10] can be seen (At the prototype V_{IN} is connected to +5V, R1 and R2 are 10k Ohm each).

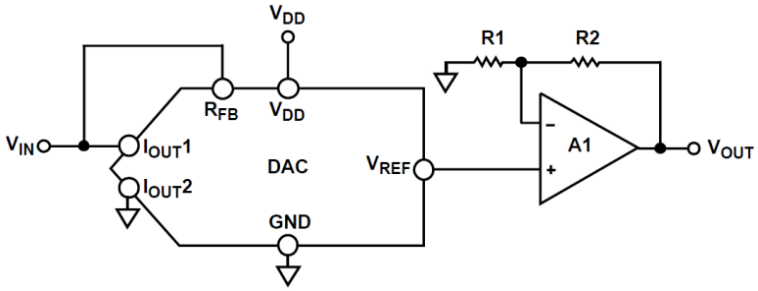


Fig.16: Connection of DAC to impedance buffer [10]

The digital 10 Bit interface of the AD5433 does also provide a read-back function which can be helpful in some applications, but was not used in the SW as the data interface is directly connected to the softcore. To increase resolution a pin compatible version with a 12 Bit interface can be assembled as well (AD5445, see also [10]). The schematic, layout and SW were prepared to handle the 8, 10 and 12 Bit version (for the prototype the 10 Bit version was assembled).

Beside the parallel data interface there are only two more lines necessary to control the DAC. One CS and one R/W signal. The R/W signal is used to configure read or write mode of the input latch. The CS signal can be seen as a clock line. At a low signal data can be written to the input latch. At rising edge of CS, data is latched and transferred to the DAC register. The DAC latches are not transparent, therefore a write sequence must include a falling and a rising edge to ensure that data is loaded into the DAC registers. The digital value is written into the DAC registers and output directly as analog value by using the R-2R network.

3.5. Analog to Digital Converter

The ADC used in the electronic is a dual 10 Bit analog to digital converter from Maxim (MAX1183). It uses a nine-stage pipelined architecture (see Fig.17) which allows high speed conversion. The input voltage is “stored” by a “track and hold circuit” at the input and processed through the pipeline. As it is shifted to the next stage every half clock cycle the total latency of the conversion is 5 clock cycles. Each stage consists of a one and a half bit flash ADC to convert the input voltage of the stage into a digital code. A digital to analog converter in each stage converts the digital result back into an analog voltage, which is then subtracted from the input signal. Before it is forwarded to the next stage the resulting voltage is multiplied by two to match the next input stage.

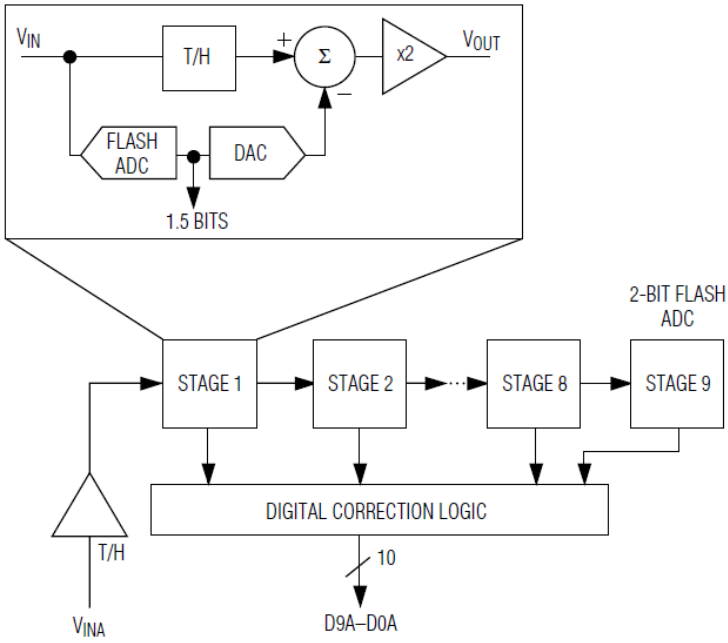


Fig.17: Nine-stage pipelined architecture of ADC [13]

The full-scale range is dependent on the configuration of the chip. For the setup of the prototype the "Internal reference mode" (see datasheet [13]) was used, by connecting the REFOUT to the REFIN pin with a 1kΩ resistor. The resulting full-scale range in this configuration is the difference of ($V_{DD}/2+V_{REFIN}/4$) and ($V_{DD}/2-V_{REFIN}/4$). As V_{DD} is 3,3V and the internal reference of the ADC is 2.048V the full-scale input range is 2,276V.
(The full-scale input values here correspond to the peak to peak value of the input signal, not to the RMS value)

Same as for the other components, also for the analog to digital converter, the chosen IC is available in other pin compatible versions, which allow higher performance. The version assembled on the prototype is a MAX1183 (10Bit 40Msps). Other versions like the MAX1182, 1181, 1180 or 1190 allow sampling rates up to120Msps at 10Bit, but do have higher prices.

3.6. Analog input with Voltage Controlled Amplifier

The **V**oltage **C**ontrolled **A**mplifier used in the circuit is a VCA2614. It has two analog channels including an input buffer, a voltage controlled attenuator and a programmable gain amplifier at each channel (see Fig.18)

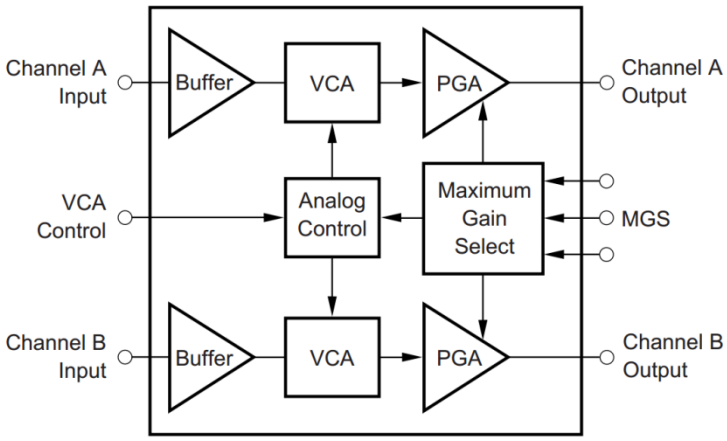


Fig.18: Block diagram of dual channel VCA [14]

All internal blocks are AC-coupled. The coupling into the PGA stage requires an external capacitor if frequencies below 75kHz should be passed on to the PGA. By assembling a 10nF capacitor the usable bandwidth of the VCA was set from 40kHz to 40MHz (the output PGA rolls off at around 40MHz).The three MSG bits do not only select the gain of the PGA but also select the maximum attenuation of the VCA. The MSG therefore selects the overall gain range, while the VCA control voltage defines the actual gain as an ideal dB-linear transfer function (see Fig.19)

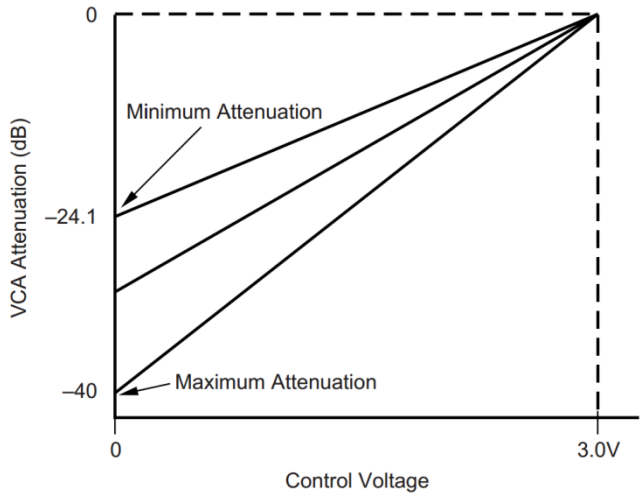


Fig.19: Swept Attenuator Characteristics [14]

For each selected digital gain code the maximum VCA attenuation (selected by VCA_{CNTL}=0V) is the inverse of the selected PGA gain. Therefore at an input voltage of 0V the VCA + PGA gain is always 0dB, independent from the digital selected gain.

As a result of this the overall gain per channel is selectable by the MGS settings and the analog input voltage. Fig.20 shows some plots of the gain as a function of the input voltage at different MGS codes.

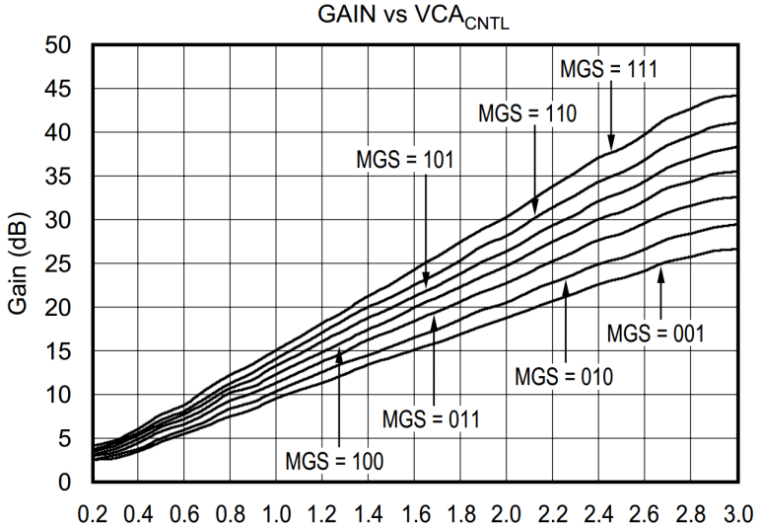


Fig.20: Plot of Gain vs. control voltage at different MSG[14]

At the prototype the gain control voltage can be either selected to be adjusted by a potentiometer, or set to the output of a buffered C charging circuit which is triggered by the start of the measurement. In case of using external microphones this allows to compensate some part of the propagation attenuation of the ultrasonic signal. As the output of the IC provides a differential signal it matches perfectly to the input of the analog to digital converter. A first order low pass filter with a cut off frequency of 21Mhz was used to avoid aliasing effects. The input is single ended and therefore can be directly connected to a single ended signal source (like the ultrasonic microphones).

To avoid unwanted distortion signals, frequencies below 40kHz were cut off by using a high pass filter at the input. In combination with the AC coupling capacitor at the PGA, a band pass characteristic was achieved in total with an attenuation of 40dB/decade at lower frequencies and a cut off frequency of 40kHz. The attenuation at higher frequencies is 20dB/decade with a cut off frequency of 21MHz.

The complete VCA2614 is supplied with 5V. Therefore the digital interface was assumed to expect also 5V digital signals (digital interface levels are not mentioned in the datasheet). To ensure proper functionality a 3V3 to 5V level shifter (74LVX4245) was used to match the logical voltage levels between the FPGA operating at 3V3 and the VCA operating at 5V.

3.7. Programmable VHDL Hardware

The exact configuration of the VHDL hardware defined by the configuration of the IP cores as well as the VHDL source codes were asked not to be disclosed in this thesis. Nevertheless the reader should get a good overview about the system and the approach of resolving the requested problem.

The programmable VHDL hardware (defined by the VHDL code and the ISE project setup) was the most time consuming part of this thesis due to two reasons:

- The Xilinx development tools allow designing highly complex systems, but starting without any training it is a long way to get familiar with all necessary settings and to get a 3rd party hardware running in stand-alone mode.
- The high variety of IP cores, which should be chosen wisely at the beginning to achieve the required performance.

The Xilinx development tools contain several programs to setup and maintain a project including simulation, optimization, debugging, programming and versioning (Xilinx ISE v14.3 was used for the development of the software of the prototype). There are a lot of tutorials available in the internet, which allow new users to get a simple VHDL code running within a few clicks (e.g. [15]). The drawback from my point of view is the high number of different possibilities to put a simple VHDL code into practice in the Xilinx ISE, resulting in a huge number of “HowTo’s and Tutorials” looking all very different. Another problem is the limitation of some tutorials to implement only some VHDL code on a specific target HW by using dedicated programming tools. The implementation of pre-defined IPs, custom IPs and a softcore running in stand-alone mode without any PC connection was only possible with the help of the online support from the Xilinx and ZTEX team.

As mentioned in chapter 2 “Concept”, a very first approach for the setup was to use a physical external microprocessor with a dual ported memory for connecting the RAM to the microcontroller and the FPGA. By changing the concept to an integration of the microcontroller into the FPGA as a softcore, the question arose: How to access the memory? As the FPGA evaluation board from ZTEX was the most promising board to get good performance at reasonable costs, the physical memory chip was defined. The memory controller inside the FPGA was the only variance remaining. Coming from a first idea of a discrete setup with the memory attached directly to the controller and missing the second physical memory interface to the FPGA due to HW constrains, the self-evident concept was to use a memory controller to attach the RAM to the microcontroller, and DMA to transfer the data to the RAM at the desired speed. In this configuration, the data interface to the external ADCs is realized by IO pins of the softcore, same as for the interface to the DAC.

The DMA configuration is well known and used in many applications to transfer high amount of data to and from the memory. This setup was put into practice with reasonable time effort and seemed to be performing well at the first tests. A closer look and tests with higher data rates soon revealed that the data transfer from the IOs of the softcore to the memory by using DMA was not sufficient to achieve the required data rates.

After a lot of investigations on the available memory controllers, a multi port memory controller (mpmc_v6_06_a see [16]) seemed to be the best solution to integrate a multi port configuration in the FPGA. For the correct interfacing a custom IP was necessary to provide the direct connection between the external ADC's and the DAC to the memory controller. It was also obvious that the clock signals cannot be provided by the softcore and have to be derived from the main clock. To get selectable clock frequencies, the custom IP was extended by a clock processing function, to provide programmable clock frequencies. The 24MHz crystal assembled on the ZTEX FPGA board is used as a common clock source for all functions.

3.7.1. MicroBlaze Softcore

The softcore which was used for the prototype was a Xilinx MicroBlaze [17]. It is a 32-Bit-RISC processor providing features which can be configured, activated or deactivated before synthesizing [18]. Some of the main features are:

- Floating Point Unit
- Memory Management Unit
- Instruction and Data Cache
- 3 to 5 Stage Pipeline
- Barrel Shifter
- Integer Multiplier
- Integer Divider
- Pattern Comparator
- Hardware Breakpoints
- JTAG control via debug support core
- Interrupt signaling

3.7.2. Peripheral IP's

Additionally to the list of functions of the MicroBlaze, several peripherals had to be implemented to get similar functionality like an external microcontroller. These IP's are not directly located inside the MicroBlaze as they are individual IP blocks, but can be seen as

part of the microprocessor inside the FPGA. In Tab. 5 those IP blocks are listed including the version number used for the prototype.

IP Block	Version used in the SW
UART interface	xps_uartlite_v1_02_a
GPIO's	xps_gpio_v2_00_a
Timer	xps_timer_v1_02_a
Interrupt service controller	xps_intc_v2_01_a
Reset controller	proc_sys_reset_v3_00_a
BRAM controller	bram_block_v1_00_a
MicroBlaze debug module	mdm_v2_10_a
LMB BRAM Interface controller	lmb_bram_if_cntlr_v3_10_b
Clock generator	clock_generator_v4_03_a

Tab. 5: MicroBlaze peripheral IP blocks

3.7.3. Multiport Memory Controller

The MPMC (mpmc_v6_06_a) from Xilinx [19] can be used to access the attached memory with several different interfaces [20]. The configuration of the physical memory interface itself was done according the datasheet t[21] of the DDR2 memory chip that is assembled on the ZTEX FPGA board. Two ports were enabled to access the memory from the softcore and from the custom IP independently from each other. The interface to the softcore is a standard PLBv64 (XILINX Processor Local Bus [22]) interface running at a bus width of 32Bit. This interface is just connected to the PLB bus used in the setup to communicate with the other IPs (no dedicated PLB bus was used). Although it is not the fastest bus available, it is fast enough to transfer the data from the PC to the memory and vice versa, as the bottleneck of the data transfer in this case is the UART. In Fig.21 a block diagram of the used configuration can be seen.

The big advantage of this configuration is the direct read/write access to the memory. The MicroBlaze can easily access the data, by accessing the mapped memory address. The second port was configured as NPI (**N**ative **P**ort **I**nterface) at a bus width of 64Bit (see datasheet [16]). Using this port configuration enables a very low-level direct access to the memory controller core, resulting in very high data rates but requiring also more complex configurations to control the interface.

The clock rates of the PLB and NPI bus were set to 75MHz, but can be increased to higher values if necessary. Due to constrains of the MPMC, the NPI must run at the same or at double frequency of the PLB bus. Therefore at double rate frequency, data rates of close to

10GBit are possible, even without changing the PLB frequency. In this case the DDR2 memory would be the bottleneck of the system.

The PHY memory clock frequency of the prototype was set to 300MHz. When using other boards than the 1.11cfrom ZTEX, also faster memory components are available (DDR3). Those still can be operated at their maximum clock frequency by using the architecture described in this thesis.

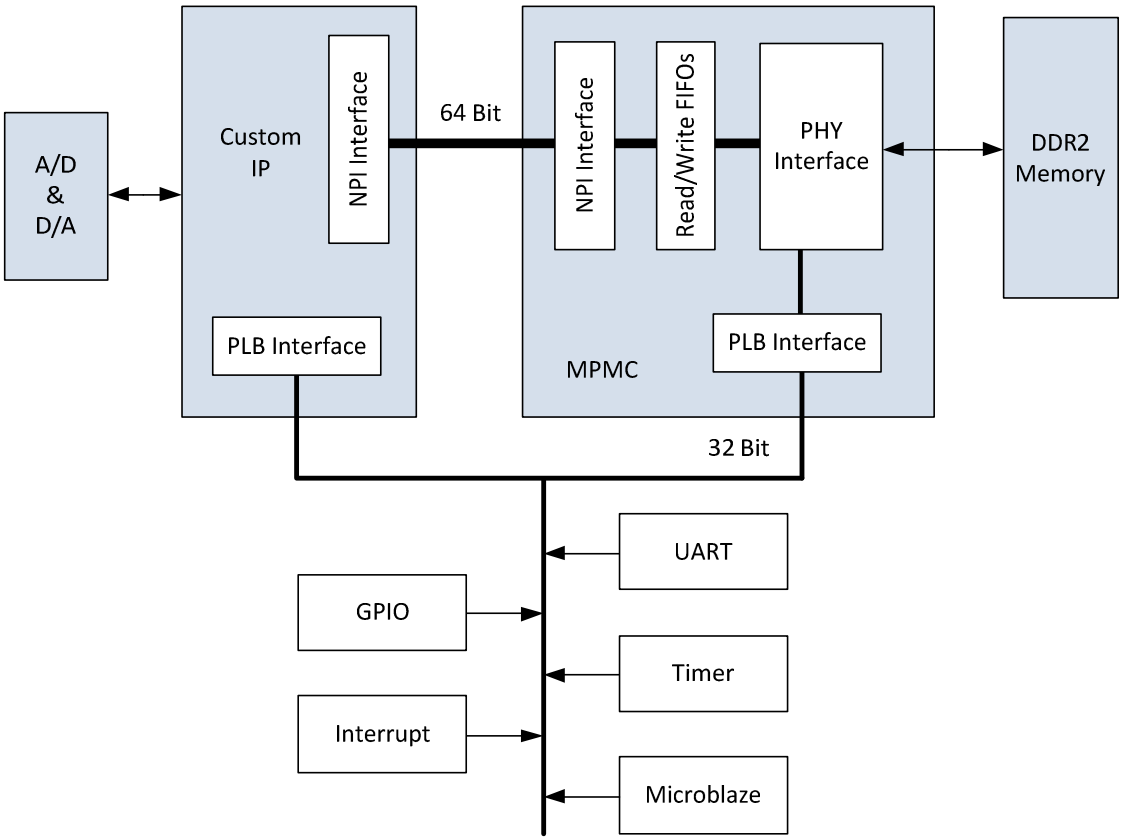


Fig.21: FPGA IP bus structure

3.7.4. Custom IP

The custom IP is a custom configured hardware part programmed in VHDL. Although it is really “programmed” during each start-up, it is mentioned here in the hardware chapter, as the code is only describing the behavior of the hardware.

The main functions are:

- Provide configurable clock signal for the external ADCs
- Provide configurable clock signal for the external DAC
- Transfer data from RAM to DAC
- Transfer data from ADCs to RAM

The above mentioned functions seem to be simple, but when using the NPI to access the MPMC things become more complex.

The clock signals for the DAC and ADC have to be generated in a defined sequence. As the ultrasonic application is based on very precise time of flight measurements of echo signals, the timing of the transmitted and received signals has to be very precise. It was a requirement to be able to adjust the length of the transmitted signal as well as the length of the recording of the echo. The recording of the echo received by the microphones and digitized by the ADCs has to start exactly after the last clock cycle of the DAC which generates the transmission signal. The duration of the transmission is defined by the number of samples divided by the sampling rate (same for the duration of the recording). The configuration of the duration of the clock signals and the clock frequencies have to be provided by the softcore (interface see chapter 4.2). The generation of the correct output clock was realized by a counter, counting the edges of the clock signal provided by the “Clock Generator IP” (see chapter 3.7.2). It is set back to zero every time it reaches a compare value. At each reset to zero the clock output from the IP toggles. The compare value for the counter is configured by the softcore. As the IP only gets a compare value, the input clock speed has to be defined also in the C code of the softcore to allow calculation of the compare values for the desired output clock frequencies. Within the test software this input clock frequency was set to 75MHz.

To signalize the start of a measurement, the trigger signal is used. The trigger signal is an output GPIO of the softcore. It is set by the softcore once the measurement command is received from the MATLAB interface. At a rising edge on the trigger line the custom IP starts to load the internal data buffers. Once the buffers are ready loaded, it starts to count the input clock and provides the required DAC output clock. In parallel the data is loaded on the parallel interface on every clock edge. The number of clock cycles at the output is again counted and monitored. When reaching a certain value (compare value provided by the softcore) the clock output of the DAC is stopped. At the last rising edge of the DAC clock output, the output for the ADC clock is started using the same principle. This time the data is read at every falling clock edge and stored to the internal buffers. Once the defied number of samples for the ADC is reached the IP signalizes the softcore the successful data transfer (see chapter 4.2).

To ensure correct routing of the clock signals inside the FPGA all clock signals were declared explicitly as “clock signals”. A clock forward technique (ODDR2 buffer) was used to forward the clock signals from the clock generator to the custom IP and also to the external components (see [23]).

The DDR2 RAM stores the information in form of small charges which can be charged or discharged, representing 1 or 0. As these charges are volatile, the complete RAM needs to be refreshed continuously even if there is no modification to preserve the information stored. This refreshing is done automatically by the MPMC. The necessary information for correct refreshing timing is provided by the datasheet of the DDR2 RAM IC [21] and was set at the configuration of the MPMC IP. During refreshing of the RAM cells there is no reading or writing possible to the data inside the RAM cells. The MPMC is equipped with a read/write FIFO which can be used to cache some data in this case. To realize a continuous data transfer between the RAM and the external converters, some additional buffers were required beside the internal FIFO. The MPMC IP provides several data transfer sizes when using the 64 Bit NPI interface. Furthermore it is possible to choose between a “burst” or “cacheline” transfer.

For a write process at “cacheline” as well as “burst mode”, first the FIFO needs to be filled with data (see datasheet [16]). Once the FIFO is full, the MPMC starts to store the data to the RAM. The finalization of the transfer from the FIFO to the RAM is signalized by a certain signal on the NPI interface. As the refresh cycles of the RAM are independent of this write process, the time until the FIFO can be accessed again can vary. Although the RAM interface is clocked with 300MHz the required time until the FIFO can be accessed again can take some time (several cycles). Therefore a continuous data transfer to the RAM cannot be guaranteed. At a read process the situation is similar. After addressing a certain RAM area, it takes some time until the NPI interface signalizes that the FIFO is fully loaded and ready for read out, as there can be refresh cycles occurring during load of the FIFO. Again a continuous transfer of data cannot be guaranteed. Beside this, the latency time is signalized at some extra lines and needs to be awaited as well, before accessing data in the FIFO.

For the custom IP the “32-Word, Burst Read/Write” transfer mode was used. It should be mentioned that Xilinx defines a “word” as 32 Bit. Therefore a “32-Word, Burst Read/Write” transfers from or to the FIFO on the 64 Bit bus requires 16 clock cycles.

To ensure continuous data transfer, additional buffer stages had to be implemented. The reload strategy was kept very simple and is described on the next pages:

Once a positive edge is seen at the trigger line, and no data transfer is processed (positive edge on the trigger line would be ignored in this case), the start of a measurement is indicated. First data from the RAM needs to be forwarded to the external DAC (cached by the FIFO and the internal buffers). To load the data from the RAM to the FIFO, the correct address

needs to be provided to the MPMC on the NPI interface. The exact handling of the interface can be found in the MPMC datasheet from Xilinx (see [16]). The initial starting address is provided by the softcore. At the end of every RAM read procedure the read address is increased by the FIFO size, to access the correct data at the next read process.

After the read from the DDR2 RAM the data is forwarded from the FIFO into the buffer which has double size of the FIFO (256 Byte). This copy process takes place at the NPI clock rate of 75MHz with 64Bits transferred per clock cycle (64 Bit bus width of NPI). The read out of the buffer is done at the desired DAC clock speed and it is therefore independent from the writing process of the buffer. After the trigger signal indicates the start, first the complete buffer is filled with data. Once the buffer is full, the DAC clock cycle is started and the data is output to the DAC. When the first half of the buffer is completely forwarded to the DAC, it is reloaded with the next data from the MPMC FIFO. During this reload process the second half of the buffer is put out to the DAC. Once the second half is completely transferred to the DAC the data is read again from the first half of the buffer and the second half is reloaded with data from the FIFO. This procedure repeats until the required number of data has been put out to the DAC. This is the case when the end address of the read process (address provided by the softcore) is reached. As the total reload time of the FIFO is much shorter than the read out time of half of the buffer, the data can be continuously transferred to the DAC at the configured DAC clock rate.

The time which is necessary to transfer data from the FIFO to the buffer at the 75MHz 64Bit NPI interface is 213ns + additional access times due to refresh cycles. The time to transfer the same amount of data at a sampling rate of 10MSPS to the DAC, is 6µs (one 12 Bit DAC data word is stored as two Byte inside the RAM). Even at high RAM access times due to refresh cycles and worse case latency times (max 3 NPI clock cycles) the buffer is filled much faster than read out again. See Fig.22 for an overview of the data transfer.

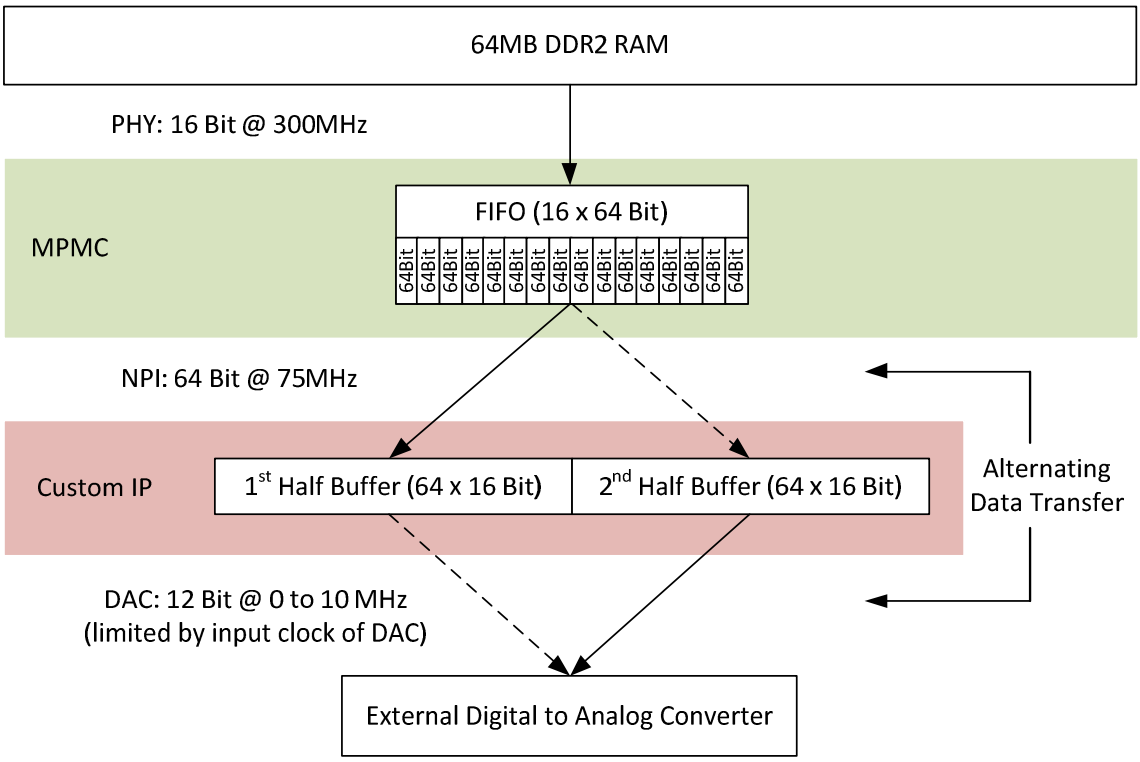


Fig.22: Buffer concept at output of data from RAM to DAC

After the last edge of the DAC clock cycle the output of the ADC clock is started. The principle of buffering is the same as used for the DAC, except the data widths and clock rates from the buffer to the ADC. See Fig.23 for an overview.

As the transfer of data from the FIFO to the buffer during RAM read is only triggered by the position of the pointer reading out the data from the buffer, the clock signals for the NPI can be completely independent to the DAC clock. The only restriction is the maximum clock frequency to ensure the reload process is finished before the dedicated buffer is accessed again. The same applies for the analog to digital converter.

As there were three clock lines implemented in the custom IP (one for each ADC and one for the DAC) even an external clock can be used to clock the data output. Minor modifications would be necessary in the VHDL code to change one of the two ADC clock outputs to an input and provide the clock for both external ADCs by one single output pin.

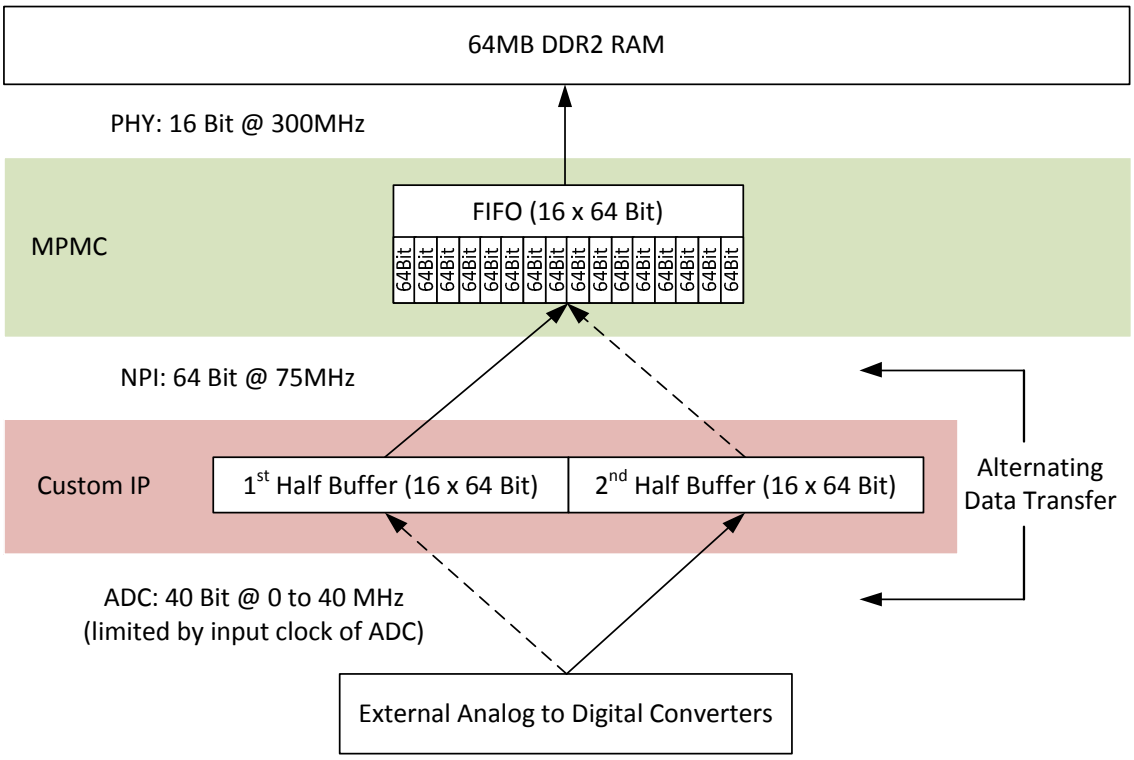


Fig.23: Buffer concept at input of data

In the following section I will take a closer look at the maximum clock rates: The clock rates used in this setting, their interrelation and possible maximum configurations are critical for successful data transfer. Therefore the individual limitations observed within the prototype shall be outlined here in detail.

The 40 Bit (4 x 10Bit) data from the external ADCs are stored as 64Bit data blocks inside the RAM. As the readout from the buffer to the RAM is done at a clock rate of 75MHz, the 40MHz maximum clock speed of the assembled ADC converter can be handled without problems. For higher data rates (at e.g. other ADCs) the NPI clock frequency needs to be increased. The clock rates of the NPI interface needs to be the same or double the clock rate of the PLB interface. The PLB interface whereas needs to be the same clock rate as the softcore, which is limited to 100MHz. The resulting maximum NPI clock rate therefore is 200MHz (at 64 Bit per clock cycle!).

Providing only 16 Bit data bus width, the bottleneck in this setup is the physical interface to the assembled DDR2 RAM which can only be increased to a clock rate of 400MHz (different

ZTEX FPGA boards provide DDR3 memories with higher clock rates, but the achieved data transfer rates are still far below the limits of the NPI bus).

The correct procedure of reading and writing according to the NPI interface specification as well as the output of the data from and to the buffer was realized in VHDL by using several state machines.

The realization shown above is the simplest form of a ring buffer containing only double buffering. For well defined data rates the number of buffers, the length of the read/write bursts, and clock rates on NPI and PHY can be changed to optimize the buffer size and consequently also the current consumption and number of cells inside the FPGA.

It should be mentioned here that the most time consuming part during the development of the custom IP was to find out how to implement a custom IP in Xilinx Platform Studio including a dedicated NPI bus to the MPMC and registers, accessible on the PLB bus for “communication” with the softcore.

4. Details on the Software Design

The software for the “Low cost high speed USB DAQ system” can be split into three blocks:

- The **MATLAB** (including the user interface)
- The softcore processor (code written in **C**)
- The **VHDL** “software” for the custom IP which was described already in chapter 3.7

The software design was also done by a top down method. First the user interface was defined including the supported commands and the protocol for each task. Afterwards the software in MATLAB and C was written to fulfill the defined functions.

4.1. Interface PC <-> Softcore (uC)

The connection between the PC software (MATLAB) and the softcore was realized by a serial interface at a speed of 921,6kBit/s (USB used as virtual COM port). To be able to test the software with a standard terminal program, only single commands were used to start the different subroutines in the softcore. After power up of the softcore the correct configuration has to be made as no default values were defined in the source code. The protocol for the configuration can be seen in Fig.24.

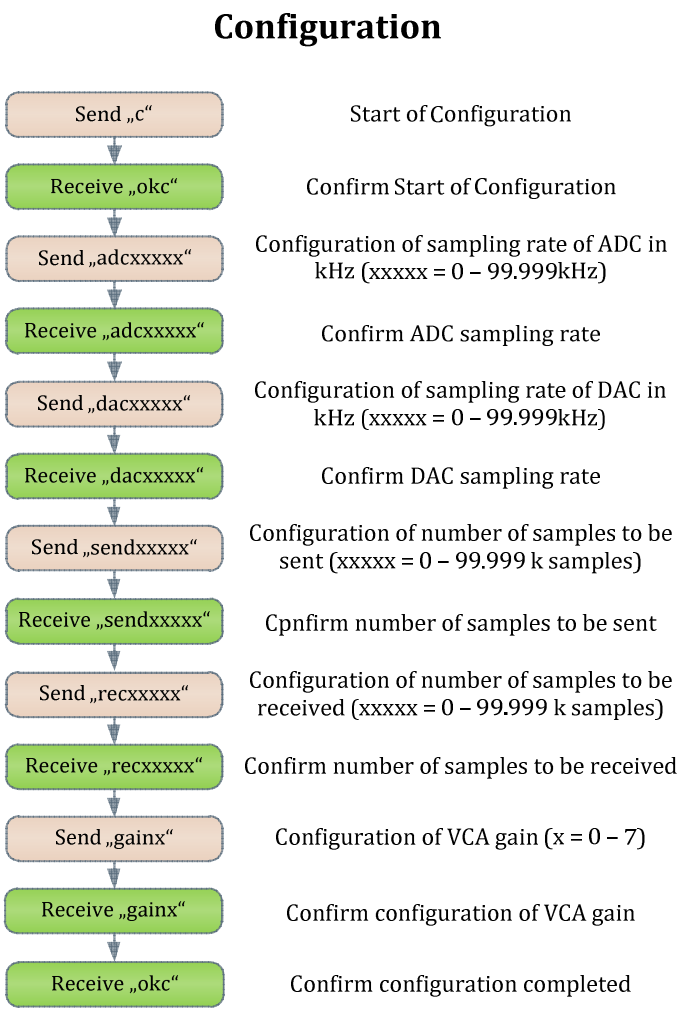


Fig.24: Configuration protocol

If any unexpected character is received by the softcore, an error message is sent back and the configuration sequence is stopped. In this case the sequence has to be started again from the beginning.

For the upload and download of data, a similar protocol was defined (see Fig.25). When using terminal programs the transfer of data was working without any problems in both directions. When implementing the software in MATLAB by using the USB virtual COM port and treating the interface as serial interface at a speed of 921,6kBit/s, problems during upload occurred. During transfer of large data, packages were lost. When receiving the data on two different PCs in parallel (split TX line of serial interface to receive data on second PC in parallel) it was observed that terminal programs do receive all data, whereas MATLAB seems to continuously lose some data packages. To avoid this problem acknowledgement

messages were implemented. As the minimum buffer size of the serial interface on a PC was found through trial and error to be at 512 Byte, the acknowledgement for upload was implemented after every 512 Bytes. At the download of data no problems were recognized, but the acknowledgements were implemented there as well to prevent a dead lock if the softcore would stop due to unexpected reasons. After implementation of the acknowledgement messages no problems were observed anymore. A drawback was some reduction of the achieved transfer speed.

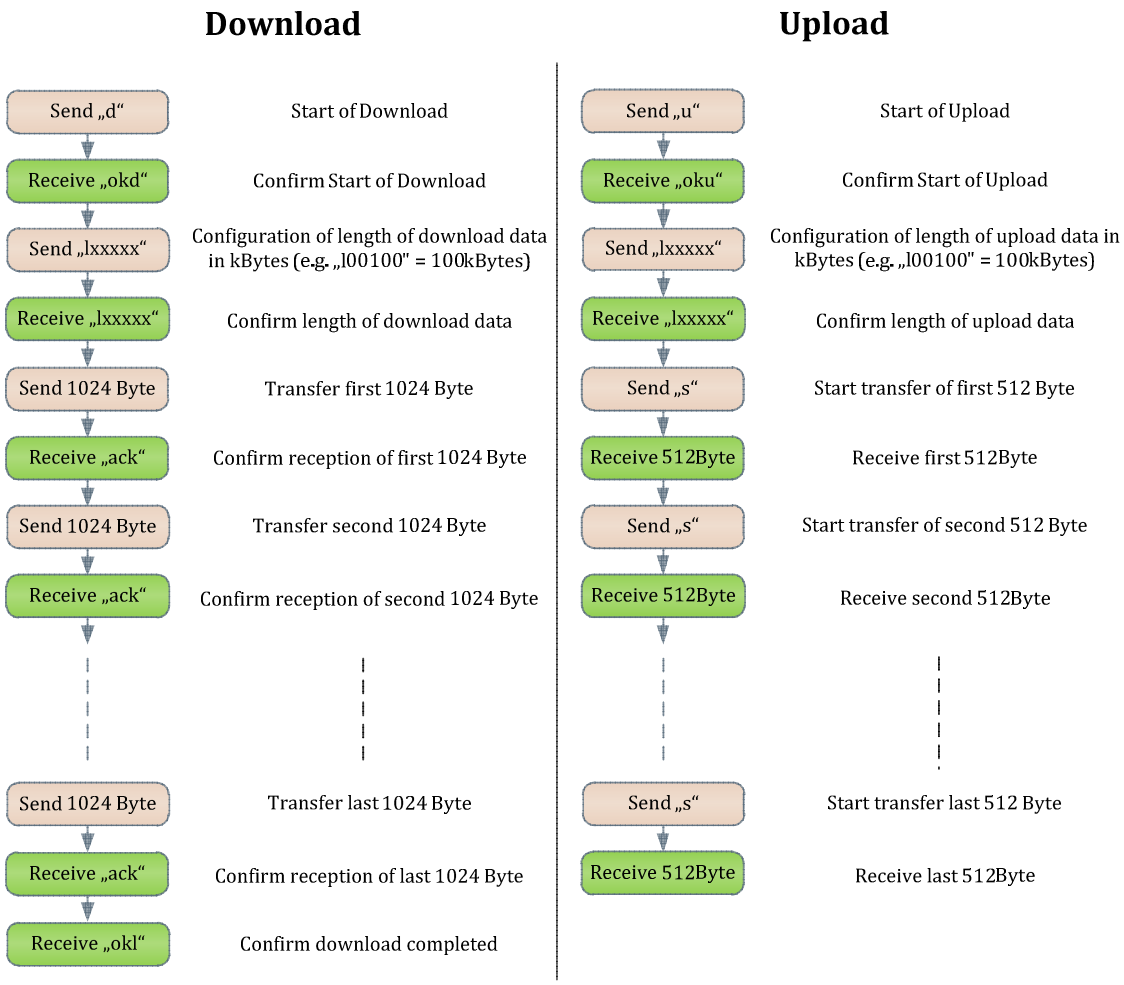


Fig.25: Up- and Download protocol

It is obvious that the length of the download data does not need to be the same length as the data to be transmitted to the DAC. Same is valid for upload data and data received by the ADC. Taking this into account, it is possible to e.g. upload only a part of the received data, analyze it and proceed with the upload if necessary. It is also possible to download a large amount of data but only output some part of it to the DAC (e.g. download several periods of a signal, but send only a burst containing a certain number of periods).

The protocol for “Power On”, “Enter Standby”, “Measure” and “Reset” can be seen in Fig.26.

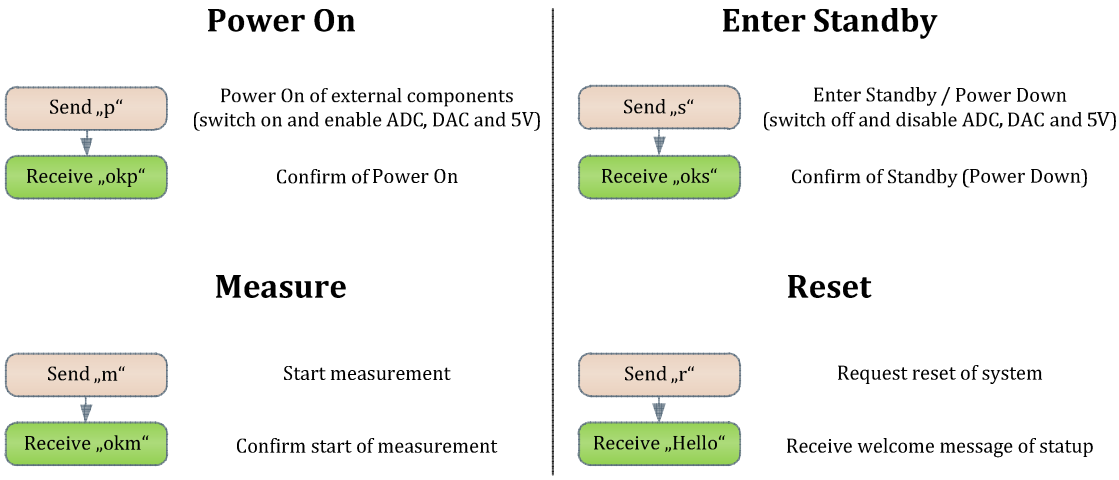


Fig.26: Power, Standby, Measure and Reset command protocol

During development some additional commands were implemented to make the debugging of the softcore and the custom IP easier.

4.2. Interface Softcore (uC) <-> custom IP

The interface between the softcore and the custom IP was realized by two HW signals as well as ten 32 Bit registers. The first hardware signal is the “trigger” signal. It is an output signal of the softcore which can also be accessed outside the FPGA (it is also used to trigger the external time dependent voltage for the voltage controlled amplifier). The second line is an input at the softcore and directly connected to the interrupt controller. At the end of a measurement this signal was planned to provide an interrupt at the softcore to enable the softcore working on different tasks during the measurement. For the prototype the interrupt signal was not used as the information on finalization of the measurement is also provided in a register. This register is polled continuously and no other tasks are pending for the softcore during the measurement. This information is sufficient for a prototype and the registers can directly be accessed on the PLB bus by any device. The function of the register can be seen in Tab. 6.

Reg.	Function	Read/Write	Comment
0	Control register	Write	Only used during SW development
1	Status register	Read	Bit 0 (LSB): ADC end address reached (active high)
2	Debug register	Read/Write	Only used during SW development
3	DAC start address	Write	32 Bit Memory address
4	DAC end address	Write	32 Bit Memory address
5	ADC start address	Write	32 Bit Memory address
6	ADC end address	Write	32 Bit Memory address
7	DAC clock prescaler value	Write	32 Bit value
8	ADC clock prescaler value	Write	32 Bit value
9	Not used	-	-

Tab. 6: Custom IP registers description

4.3. PC Software (MATLAB)

The MATLAB Interface was generated to automatically perform a complete measurement. This includes the configuration-, download-, measurement- and upload- procedure. The interface was kept simple as it is only for demonstration purpose. Inputs in MATLAB are the configuration data as well as the data of the signal to be sent (read out from an input file “download.bin”). For test purpose a sine including several periods was generated and stored into the download file. The output of the MATLAB program after a measurement are the recorded and uploaded signals provided as “upload.bin” file and plot on the screen.

The configurations can be changed in the “ultrasonic.m”. All sub tasks were split into separate files: “config.m”, “upload.m”, “download.m”, “measure.m”, “ADCDAC_power.m” and “enter_standby.m”

An example for generation of the download file can be found in the “generate_download_file.m”.

To measure the frequency response of the complete circuit an additional program was generated which periodically performs a measurement at different frequencies, measures the amplitude of the received signal and plots the frequency response of all channels. For full automation of this measurement procedure an arbitrary waveform generator from Agilent with USB interface was used. The signal generator is controlled in MATLAB by using the Agilent IO Libraries.

4.4. uC Software (C)

The software of the softcore was programmed in C. The compiler for the MicroBlaze is provided with the Xilinx Software Development Kit. When exporting the softcore hardware to the Software Development Kit, all necessary header files are generated automatically. The external DDR2 memory attached to the PLB bus by the MPMC can directly be accessed by addressing the mapped address range in the C code. The GPIOs, timers, UART and Interrupt IP's which are connected to the PLB bus, can also directly be accessed by the mapped addresses.

The test software consists of a simple state machine following the protocol described in chapter 4.1. Every data transfer for up- or download is handled by using direct access to the RAM via the PLB bus. The faster NPI interface on the MPMC is only used by the custom IP during measurements.

5. Measurements & Results

Although the electronic can be used for data acquisitions for a lot of different purposes, the ultrasonic application was kept in focus during the development. The following measurements of the characteristics do also focus on properties which are most relevant for ultrasonic sound applications.

5.1. Measurements of ADC & DAC

For ultrasonic data acquisition the performance of the analog frontend as well as the analog to digital converter are crucial. Furthermore the performance of the digital to analog converter including the output buffer amplifier is important for signal generation. All measurements in the following chapters were made at a sampling rate of 37,5 MHz for the ADC and 9,375 MHz for the DAC.

5.2. ADC Frequency Response

The frequency response is mainly dependent on the input filter, the voltage controlled amplifier and the aliasing filter at the input of the analog to digital converter. The filter cut off frequency was selected with respect to the ultrasonic application to filter unwanted acoustic signals which may interfere with the echo of the transmitted signal. The frequency response was measured at different gain levels to see if the variable gain amplifier has the same frequency response and thus gain independent phase (see Fig.27, Fig.28 and Fig.29).

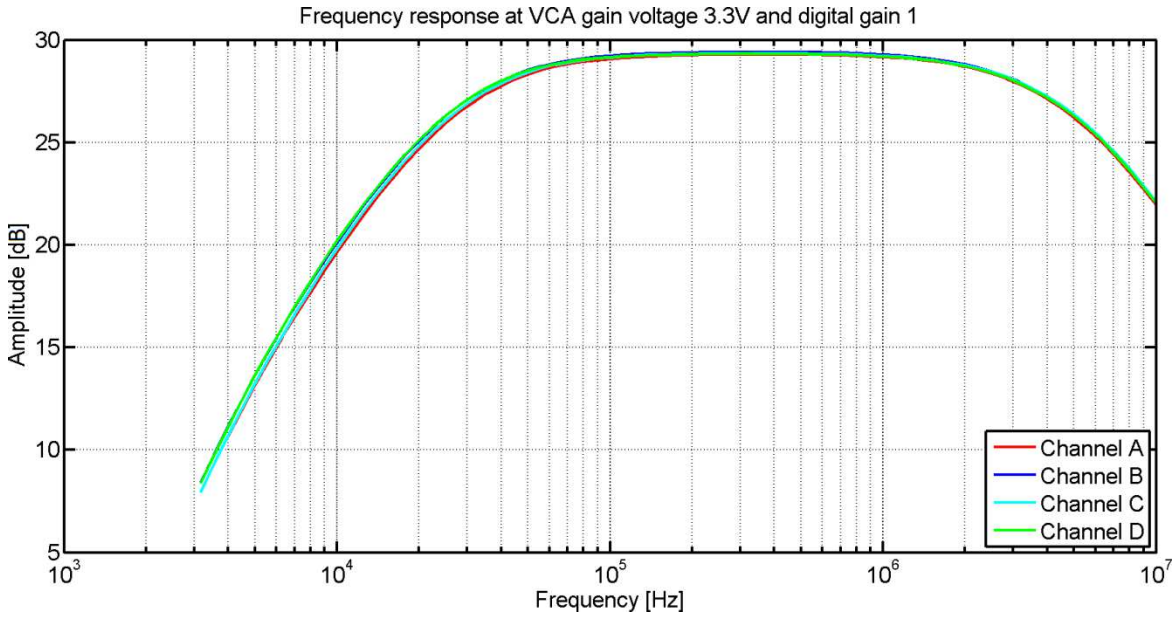


Fig.27: Frequency response of all channels (VCA=3V3, dig. gain = 1)

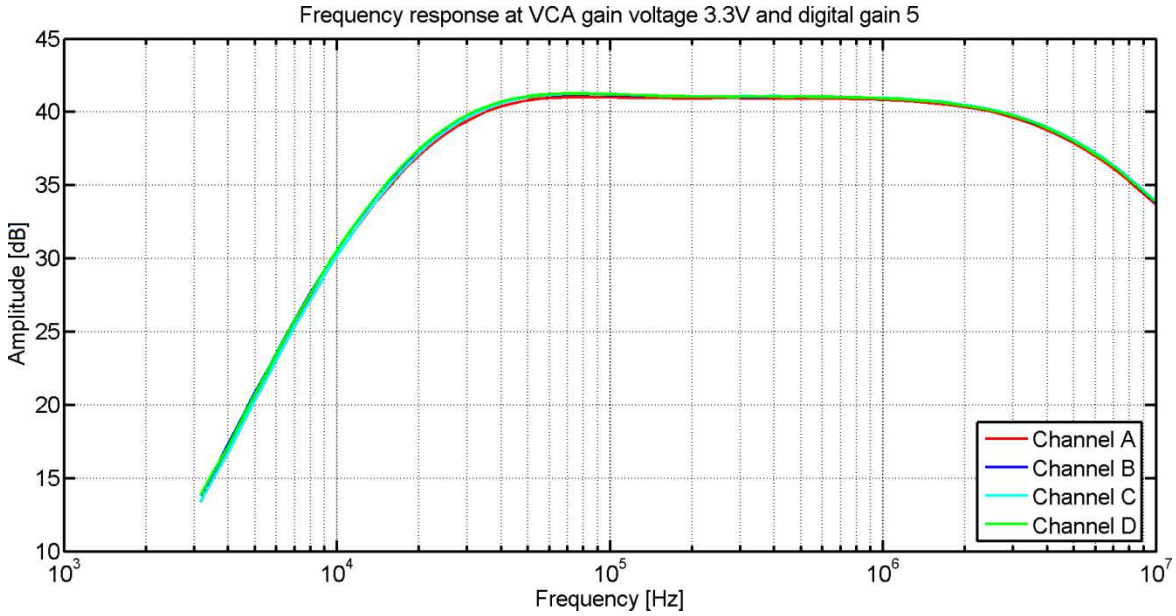


Fig.28: Frequency response of all channels (VCA=3V3, dig. gain = 5)

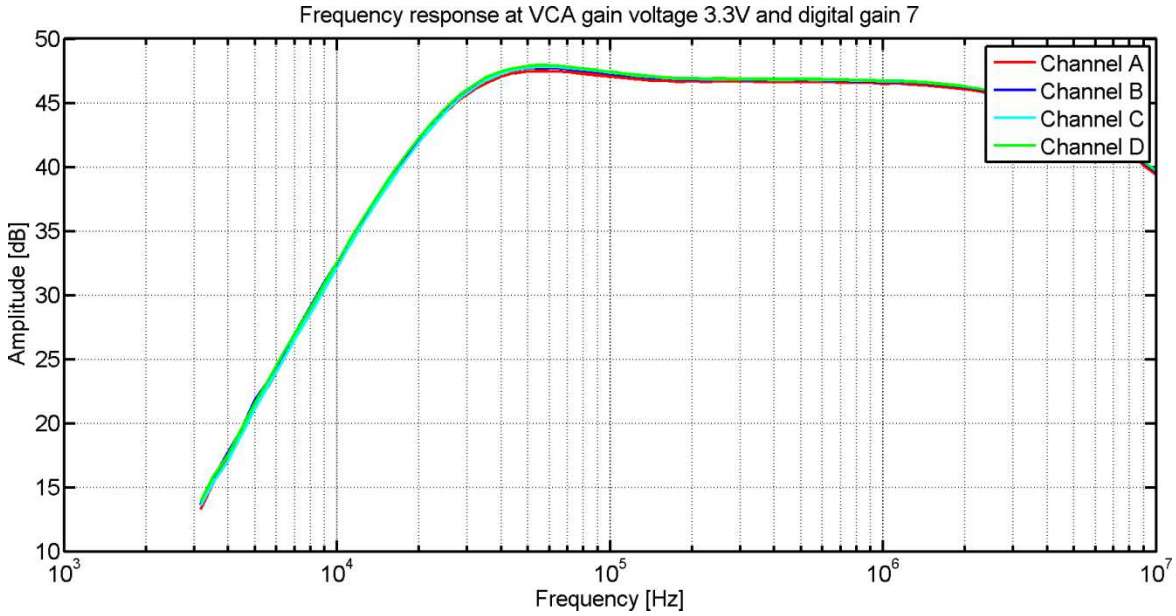


Fig.29: Frequency response of all channels (VCA=3V3, dig. gain = 7)

The plots show that the frequency response is very stable from one channel to the other, but there is a very high dependency on the digital gain selection. Especially at the lower frequency range (10-60kHz) a significant “gain peaking” appears at higher digital gain selections.

5.2.1. ADC Noise

The ADC noise was measured at different gain levels to see if the noise is caused by the amplifier or by the analog to digital converter (see Fig.30, Fig.31, Fig.32 and Fig.33).

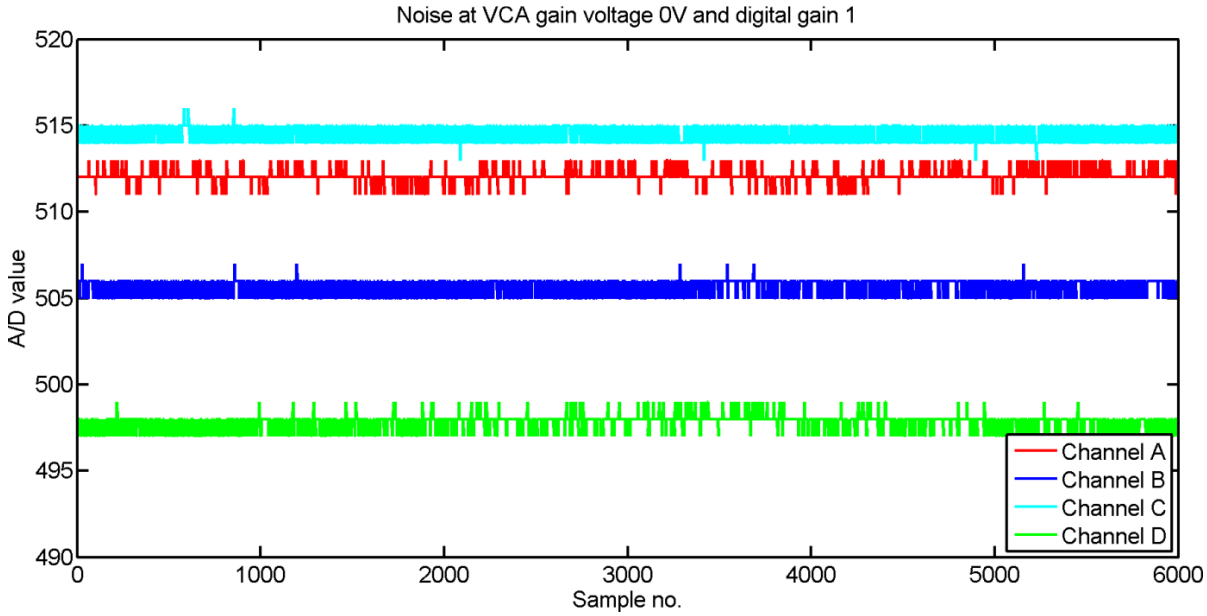


Fig.30: Measured noise at all inputs set to GND (VCA=0V dig. gain = 1)

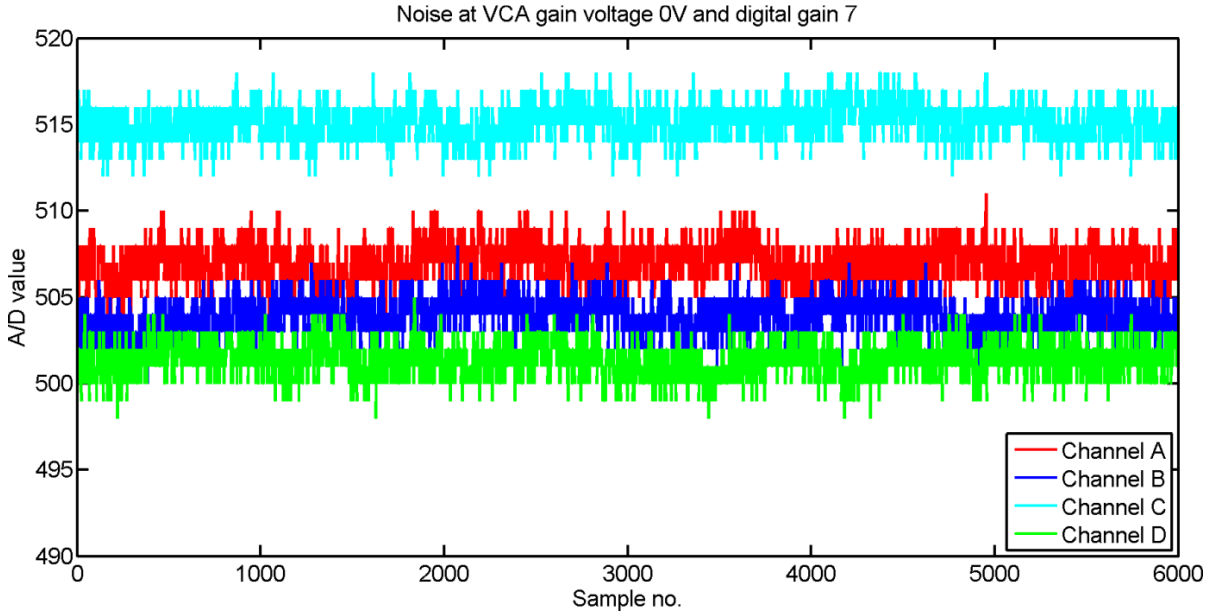


Fig.31: Measured noise at all inputs set to GND (VCA=0V dig. gain = 7)

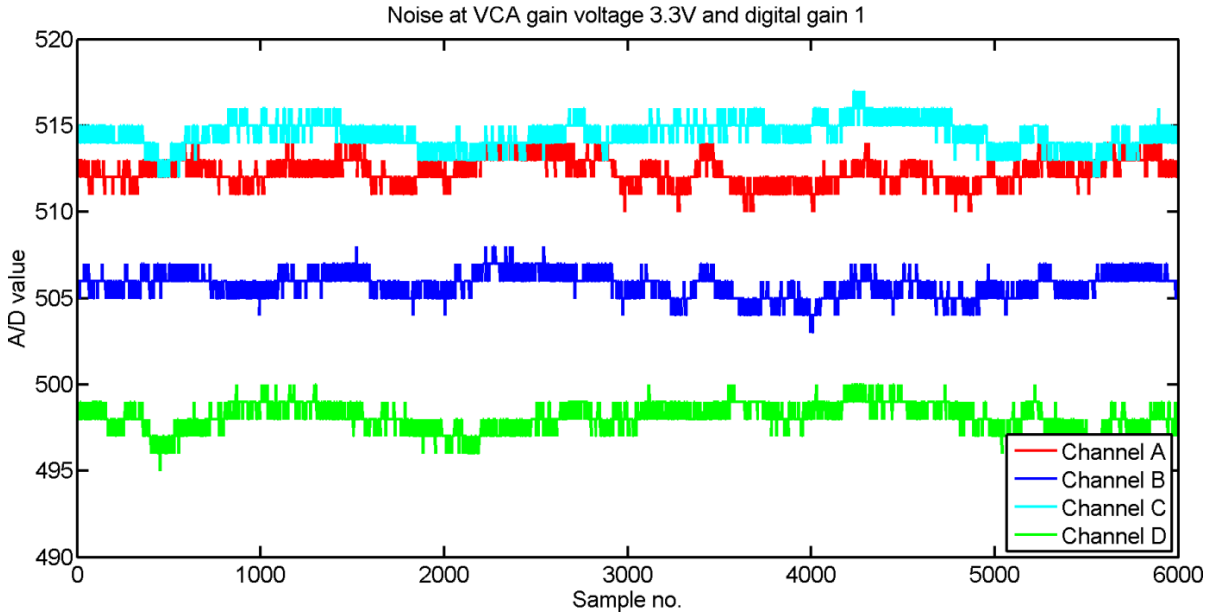


Fig.32: Measured noise at all inputs set to GND (VCA=3V3 dig. gain = 1)

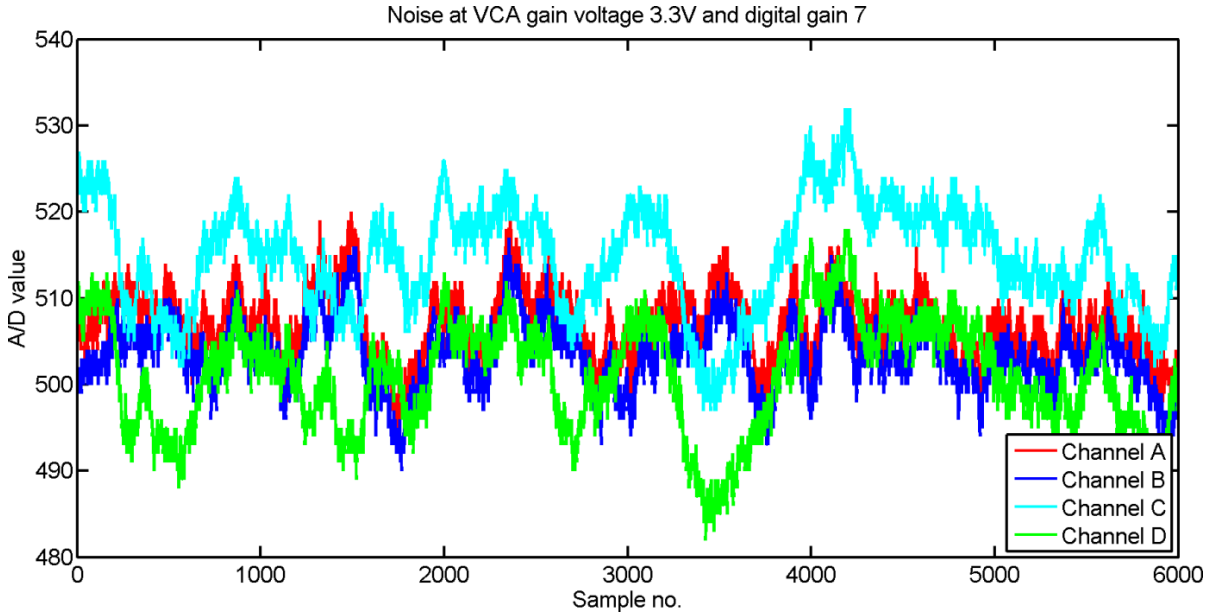


Fig.33: Measured noise at all inputs set to GND (VCA=3V3 dig. gain = 7)

As the noise is increasing at higher gain settings of the VCA, it is obvious that it is not generated by the ADC. Measurements with a high speed oscilloscope have shown that the data transfer from the analog to digital converter to the FPGA generates a lot of distortions. Fig.34 shows the record of the DAC output signal and the AC coupled supply voltage of the VCA. The block of higher noise level after the output of the DAC (sine) is appearing exactly during the transfer of the data on the parallel interface from the ADC to the FPGA. These

distortions can be found at several points in the circuit and are most probably the reason for the output noise seen at higher gain settings.

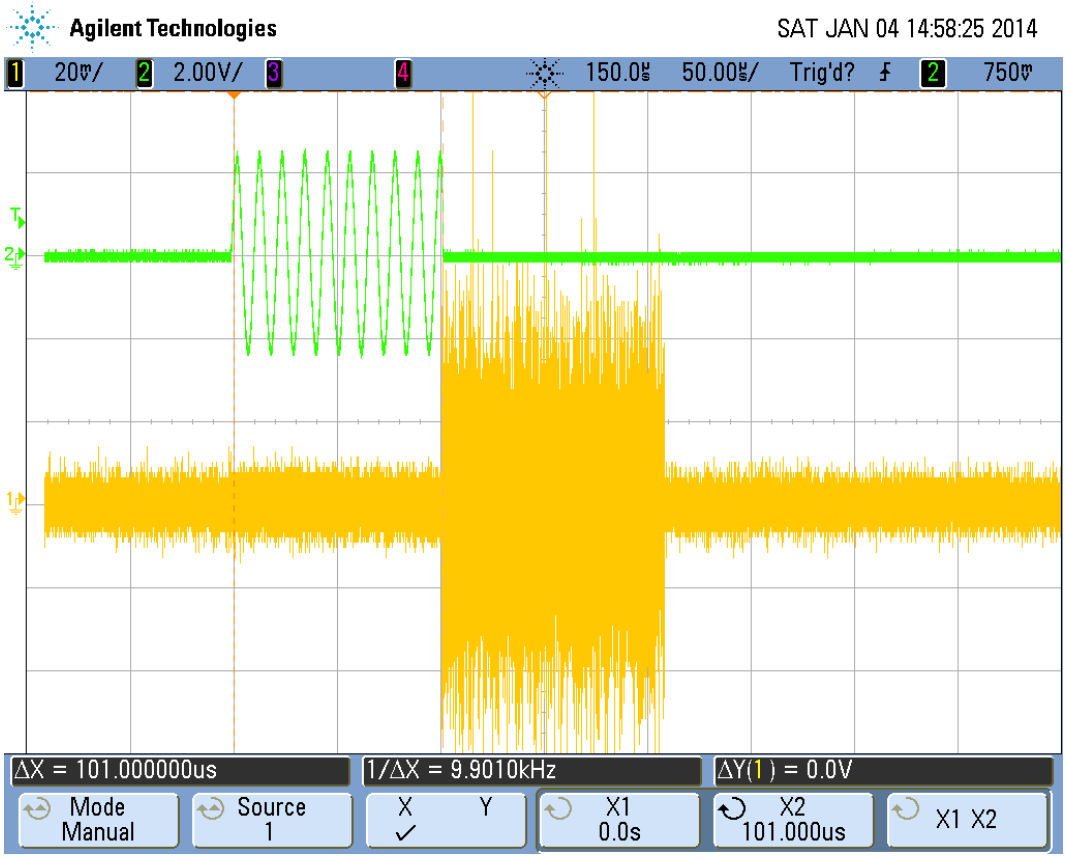


Fig.34: Noise measured with oscilloscope at GND close to ADC during sampling

5.2.2. Input Channel Crosstalk

The ADC crosstalk was measured on the complete frequency range and at different gain settings. As already observed at the frequency response measurements, a very high dependency of the selected gain was measured also at the crosstalk behavior. In the datasheet of the VCA2614 “LOW CROSSTALK: 10dB at Max Gain, 5MHz” [14] is mentioned as a feature of the chip. A closer look on the diagram at page 7 of the datasheet “Crosstalk vs. Frequency” confirmed the much lower values which were measured at different gain settings.

Following figures show the channel crosstalk at different gain configurations:

- Fig.35: Channel Crosstalk at single input signal (VCA=0V, dig. gain = 1)
- Fig.36: Channel Crosstalk at single input signal (VCA=1V, dig. gain = 1)
- Fig.37: Channel Crosstalk at single input signal (VCA=2V, dig. gain = 1)
- Fig.38: Channel Crosstalk at single input signal (VCA=3V3, dig. gain = 1)

- Fig.39: Channel Crosstalk at single input signal (VCA=0V, dig. gain = 7)
- Fig.40: Channel Crosstalk at single input signal (VCA=1V, dig. gain = 7)
- Fig.41: Channel Crosstalk at single input signal (VCA=2V, dig. gain = 7)
- Fig.42: Channel Crosstalk at single input signal (VCA=3V3, dig. gain = 7)

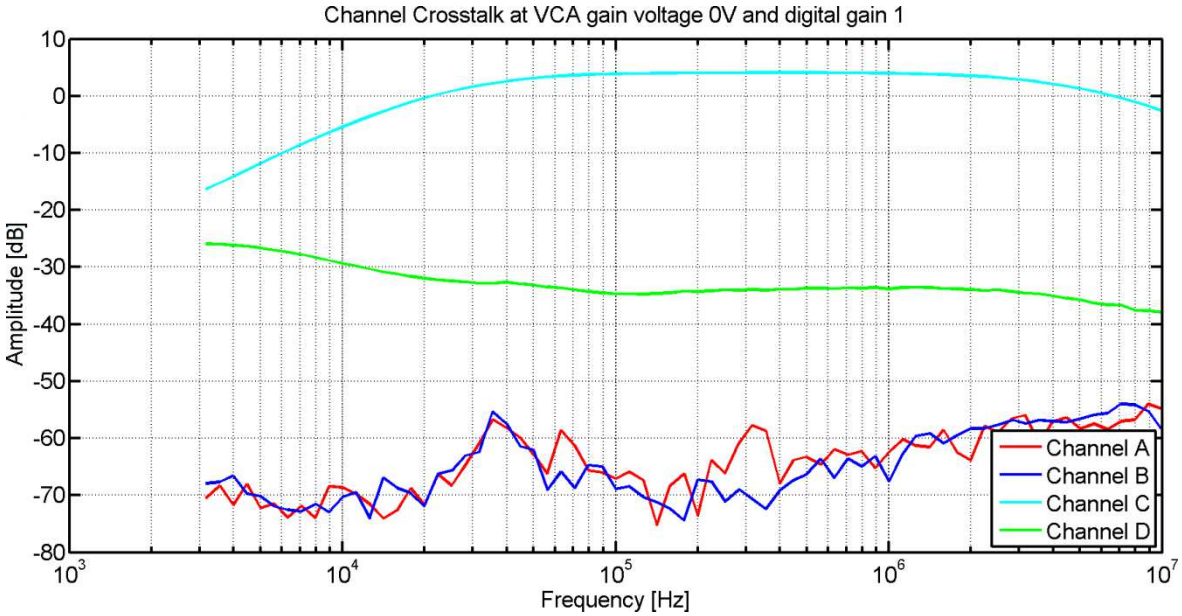


Fig.35: Channel Crosstalk at single input signal (VCA=0V, dig. gain = 1)

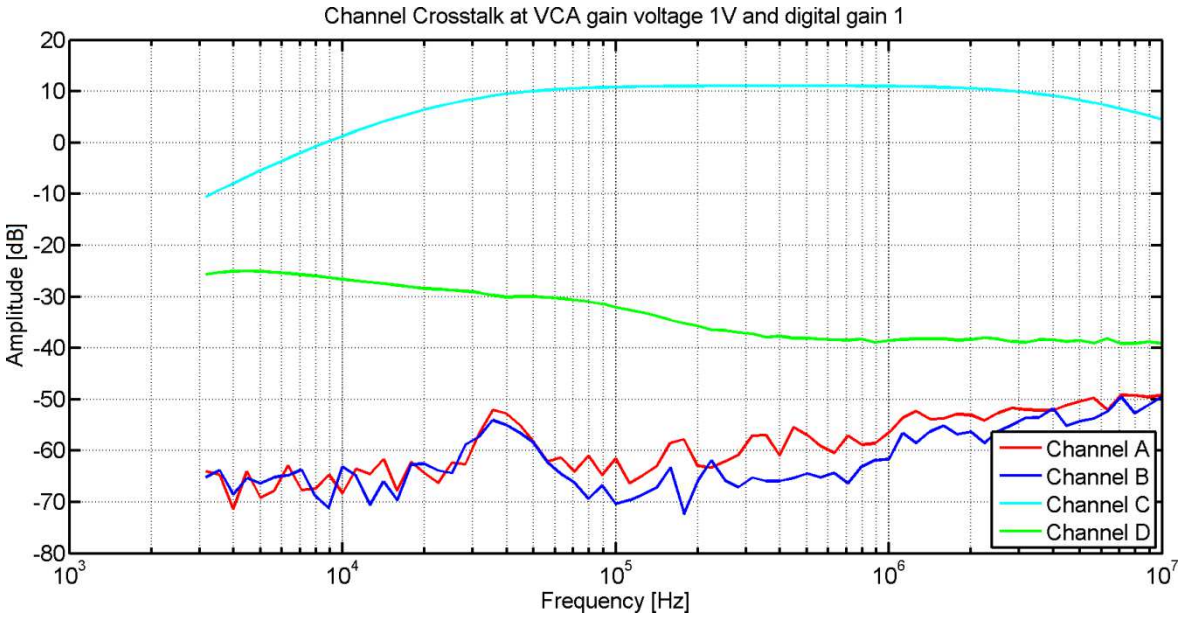


Fig.36: Channel Crosstalk at single input signal (VCA=1V, dig. gain = 1)

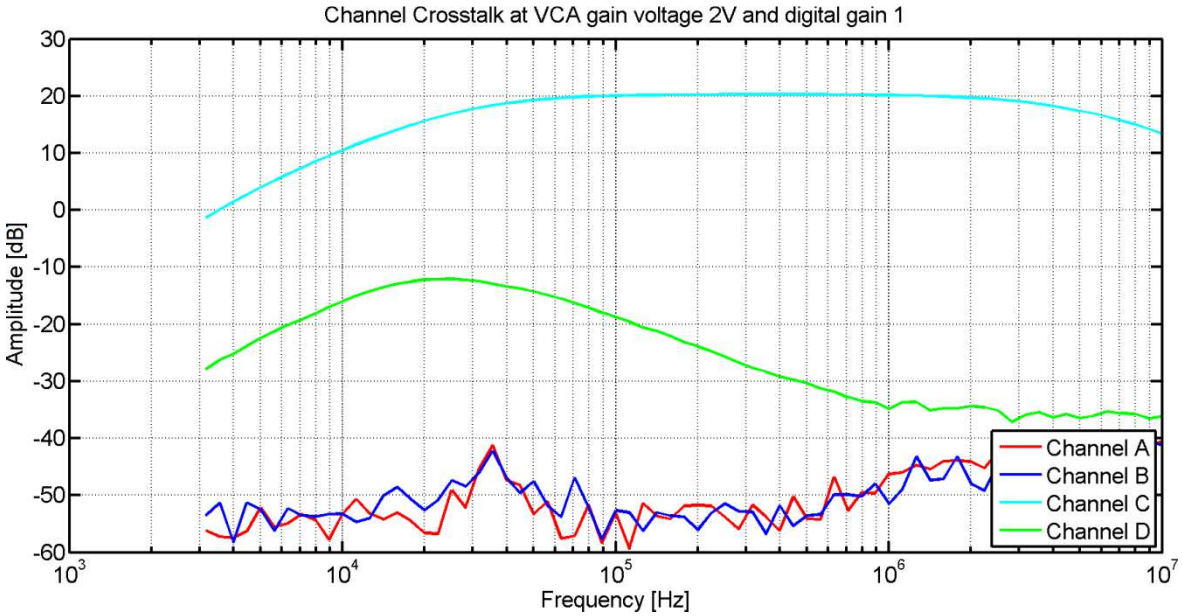


Fig.37: Channel Crosstalk at single input signal (VCA=2V, dig. gain = 1)

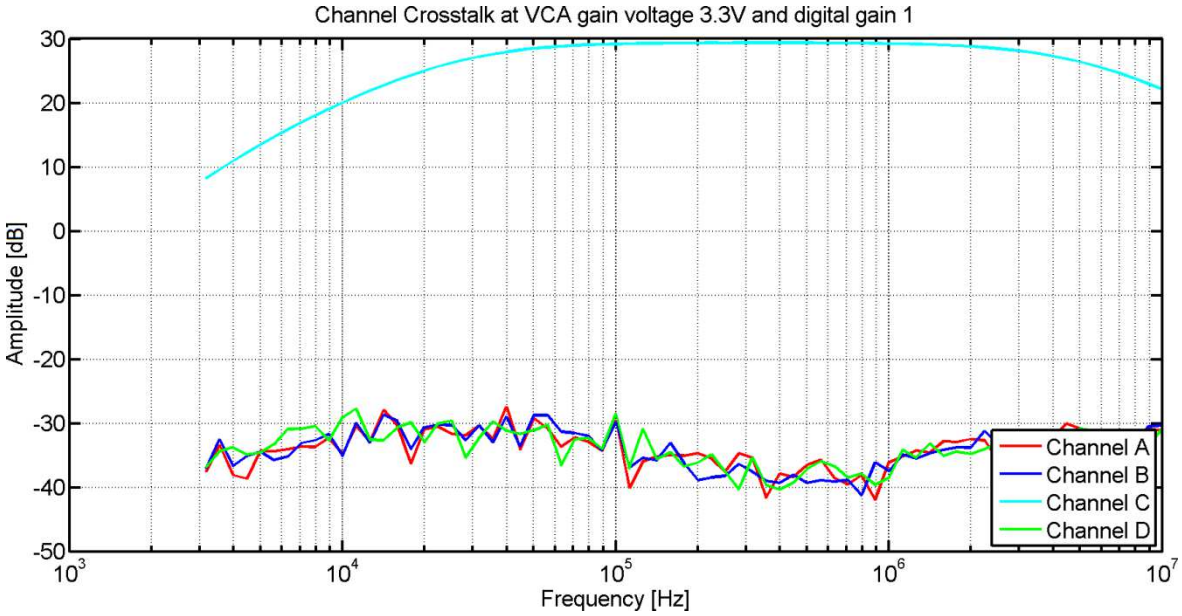


Fig.38: Channel Crosstalk at single input signal (VCA=3V3, dig. gain = 1)

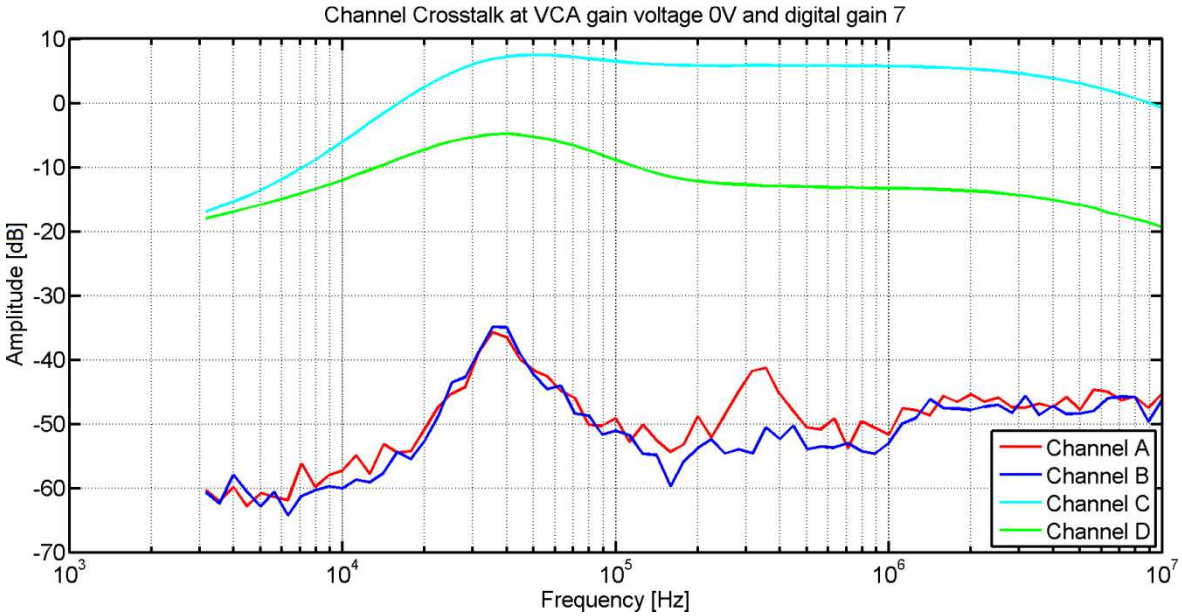


Fig.39: Channel Crosstalk at single input signal (VCA=0V, dig. gain = 7)

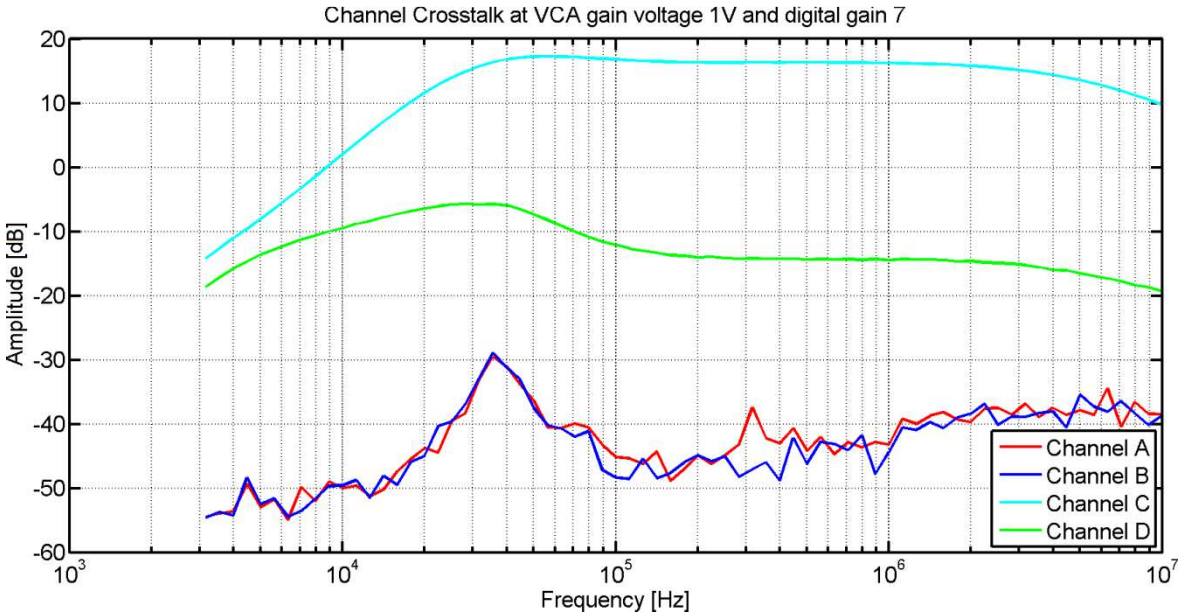


Fig.40: Channel Crosstalk at single input signal (VCA=1V, dig. gain = 7)

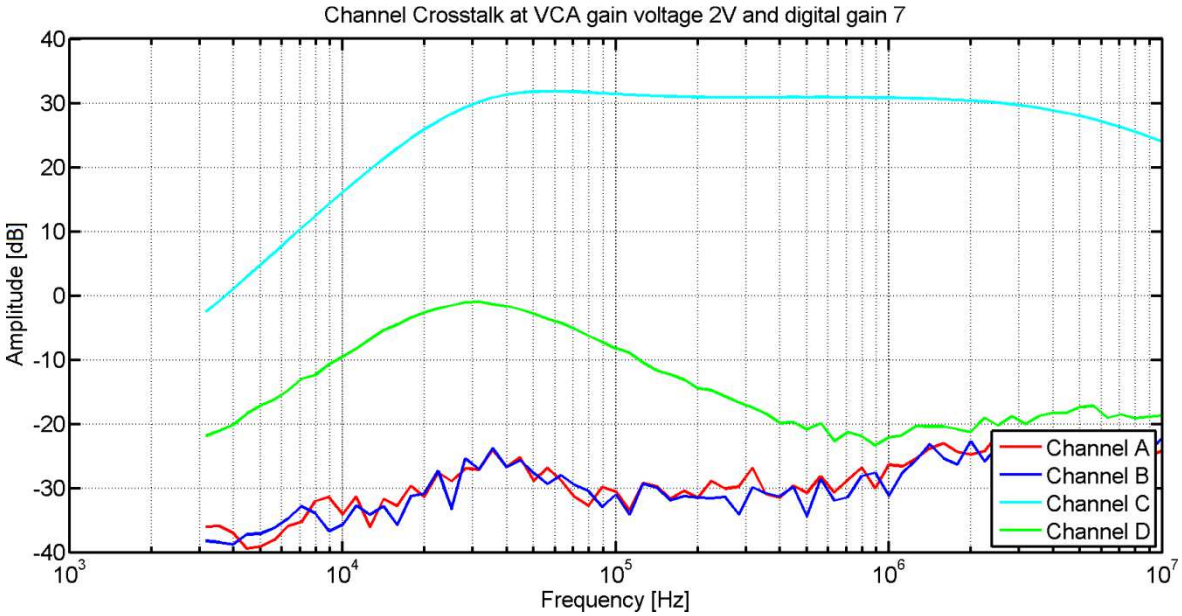


Fig.41: Channel Crosstalk at single input signal (VCA=2V, dig. gain = 7)

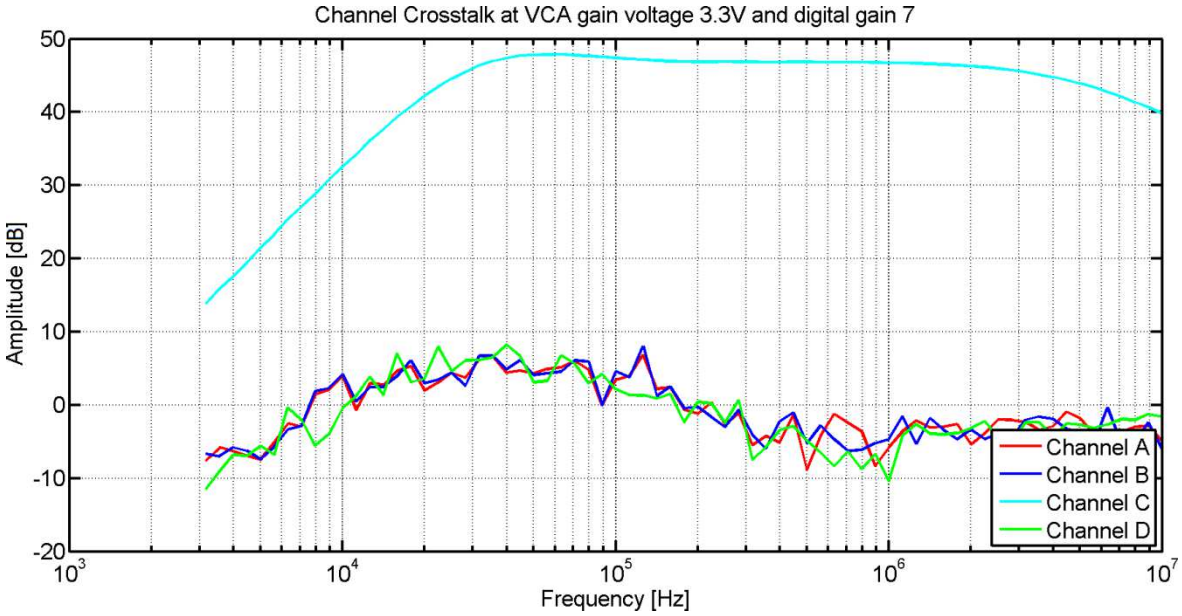


Fig.42: Channel Crosstalk at single input signal (VCA=3V3, dig. gain = 7)

5.2.3. DAC Measurements

To test the DAC a sine wave signal was generated in MATLAB. The amplitude of the signal was chosen to match the full scale of the DAC (10Bit). Fig.43 shows a typical burst signal as it can be used for ultrasonic applications.

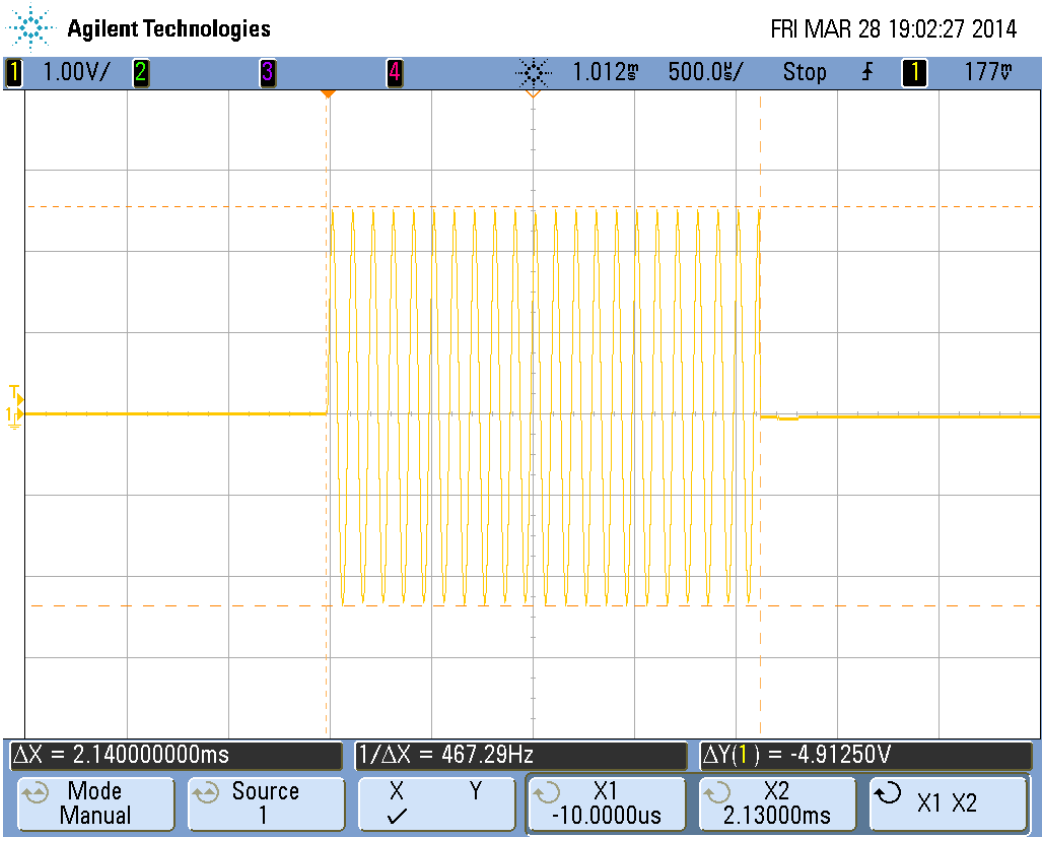


Fig.43: DAC output sine burst signal

At higher frequencies the voltage change from one digital value to the next one can be seen. See Fig.44 for an example of an output sine signal with 100kHz at a sampling rate of 9,375MHz.

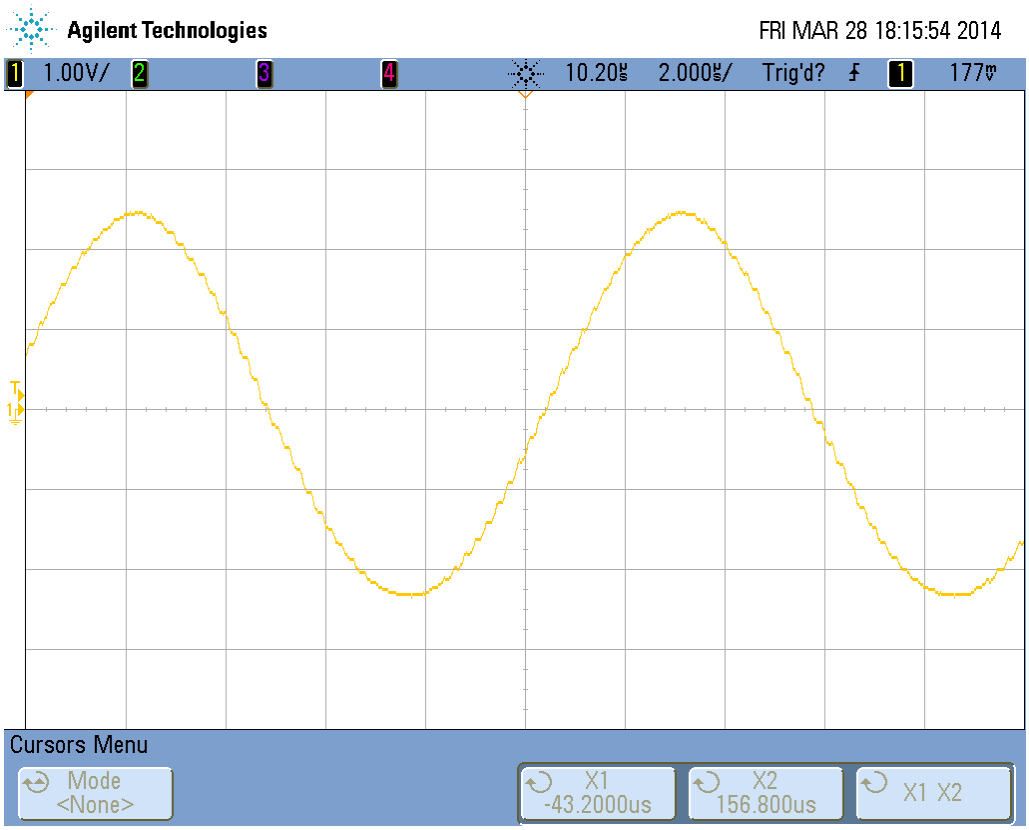


Fig.44: DAC output sine - time zoom view

5.3. Measurements of power supply

For mobile applications it is very important to know the current consumption of the electronic to estimate the load on the battery. Also for stationary operation it is important to know which power supply is sufficient to ensure the required supply current.

5.3.1. Current consumption at different operating modes

Although the current consumption of the electronic is highly dependent on the implemented functions inside the FPGA, some typical values at different operating modes and input voltages are mentioned here to get an overview about the necessary power supply for the prototype (see Tab. 7).

input voltage	current consumption @ power enabled	current consumption @ sampling of data
6.1 V	160 mA	330 mA
8 V	140 mA	310 mA
10 V	130 mA	300 mA
12 V	120 mA	290 mA
15 V	110 mA	280 mA

Tab. 7: Current consumption at different supply voltages

5.4. Data Transfer times

The data transfer times of the up- and download theoretically should only be dependent on the used transfer speed of the serial interface. As acknowledgement messages had to be inserted to ensure all data is transferred correctly during upload, the achieved overall data rate is beyond this calculated theoretical value. Therefore some measurements on the transfer speed were made with the test software on the prototype. The upload speed was measured at 112kBit/s and the download speed was measured at 208kBit/s for large data transfers.

5.5. Discussion of measurements

Exceptionally high effort was invested in the measurement of the frequency response, as this behavior is very important for ultrasonic applications. Unfortunately the results were not as good as expected. The high variation of the frequency response was not expected as the datasheet of the voltage controlled amplifier includes only information about one single digital gain selection. There is also some crosstalk between the two chips, with a maximum at approximately 40kHz (at digital gain of 7). The same maximum can be observed at the crosstalk between the two channels inside one chip. Furthermore at the frequency response of each channel a “gain peaking” can be detected at the same frequency.

As all measurements were made with input signals which use almost full scale at the pass band area, highest input signal amplitudes were applied at lowest gain selection. The chip to chip crosstalk nevertheless decreased at higher input signal amplitudes (lower gain settings). Coupling from one chip to the other on the supply and ground lines therefore seems to be low. The observed behavior indicates some resonance effects at the frequency of 40kHz, which is damped depending on the gain settings of the chip due to the different FET resistance of the voltage controlled attenuator inside the chip.

A variation of the gain settings during a measurement to compensate the attenuation caused by the propagation of the ultrasonic signals would result in unpredictable phase shifts of the received signals. Although these phase shifts would be stable from one channel to the other,

the gain dependant crosstalk still can lead to signals which are not received by the microphones but are generated as a result of the crosstalk at the VCA. The best performance was observed at a digital gain setting of 1 and an analog gain control voltage of 3.3V. In this case the crosstalk is >60dB at the frequency range of 40kHz – 1MHz, which is also suitable for ultrasonic applications.

6. Outlook& Conclusion

Despite careful planning, two mistakes were made during the layout of the prototype:

- The level shifter which is used to convert the 3.3V digital interface of the FPGA to the voltage controlled amplifier, which is operating at a supply voltage of 5V, can be set to work in both directions. Nevertheless the voltage at port A and B cannot be chosen independently. At the prototype port A and B were mixed and had to be rerouted with wires on the PCB.

- The power diodes which were used at the switching power supply did not exist in the library of the used layout software. Therefore the footprint had to be generated. Unfortunately anode and cathode were mixed up. At the PCB it was easily possible to solder the diodes also in the other direction. For the next layout this mistake can easily be reworked.

Furthermore some recommendations shall be given for future improvements:

As mentioned in the datasheet the clock lines of the ADC are very sensitive regarding distortions and should be routed separately from the data signals. Although no problems were observed, the routing recommendation may be taken into consideration on the next layout.

Another recommendation in the datasheet is to use low pass filters on all data lines close to the ADC to avoid distortions caused by the low impedance parallel data interface of the ADC to the FPGA. Although it will occupy a lot of space on the PCB, this filters should be implemented as the rise time of the ADC data outputs are very high and generate a lot of distortions. These distortion signals were measured with a high speed oscilloscope at several points on the PCB during sampling of data (see chapter 5.2.1).

Due to bad performance of the voltage controlled amplifier, another chip should be chosen for the rework of the prototype. The measurements have shown that the channel crosstalk is highly related to this chip.

Another improvement can be made at the USB interface. As discussed in chapter 4.1 acknowledgement messages had to be implemented to ensure secure data transfer. To ensure the problem is related to the acknowledgement messages, tests at different block sizes were performed. They have clearly shown that higher transfer speed rates are possible as the acknowledgement message is very time consuming at the transmission. To achieve the maximum possible data rate of 921,6kBit/s and avoid transmission errors, the virtual COM port should be replaced by the FTDI MATLAB interface. The acknowledgement messages can easily be removed in the C code of the softcore.

By reworking the software of the softcore as well as the software of the Cypress microcontroller it is also possible to change to USB 2.0 full speed (480MBit/s). In this case a higher effort is necessary as an USB interface driver needs to be prepared to connect to the FPGA board with USB 2.0.

Realizing the optimizations described here during a rework of hardware and software will ensure the functionality that is required for the use with a mobile robot as well as a lot of other applications.

The prototype which was built and tested for this diploma thesis fulfils all the requirements that where desired from the implementation. The main effort was the evaluation of the best suitable configuration in HW and SW. Especially the setup of the project in the Xilinx development environment and the implementation of the custom IP with the bus structure for high data rates were very time consuming and only possible after a deep study on manuals and datasheets about the Xilinx Development Studio and the used IP cores. All this information including source codes, project files and the working prototype was provided to the workgroup “Measurement and Control” at the Institute of Electrodynamics, Microwave and Circuit Engineering at the Technical University of Vienna after completion of the thesis for implementation with a mobile robot.

List of figures

List of literature

[1] XILINX, "ISE WebPACK Design Software," XILINX, 2014. [Online]. Available: http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm. [Accessed 12 March 2014].

[2] M. Platzner and L. Thiele, *Hardware/Software Codesign, Skriptum zur Vorlesung,* Institut für Technische Informatik und Kommunikationsnetze ETH Zürich, 2006.

[3] National Instruments Inc., 27 May 2013. [Online]. Available: http://www.ni.com/white-paper/6983/de/. [Accessed 12 March 2014].

[4] IEEE, *IEEE Standard VHDL Language Reference Manual,* 3 Park Avenue, New York, NY 10016-5997, USA: The Institute of Electrical and Electronics Engineers, Inc., 2000.

[5] "XILINX," XILINX Inc., 2014. [Online]. Available: http://www.xilinx.com/. [Accessed 12 March 2014].

[6] "Altera," Altera Corporation, 1995 - 2014. [Online]. Available: http://www.altera.com/. [Accessed 12 March 2014].

[7] ZTEX GmbH, "ZTEX FPGA-Boards mit quelloffenem SDK," ZTEX GmbH, 2014. [Online]. Available: http://www.ztex.de/index.d.html. [Accessed 12 March 2014].

[8] ZTEX GmbH, "USB FPGA-Module-1.11: Spartan 6 LX9 bis LX25 FPGA-Board mit USB 2.0-Mikrocontroller und 64 MByte DDR SDRAM," ZTEX GmbH, 2014. [Online]. Available: http://www.ztex.de/usb-fpga-1/usb-fpga-1.11.d.html. [Accessed 12 March 2014].

[9] "FTDI Chip," Future Technology Devices International Ltd., 2014. [Online]. Available: http://www.ftdichip.com/. [Accessed 12 March 2014].

[10] Analog Devices, Inc., *8-/10-/12-Bit, High Bandwidth Multiplying DACs with Parallel Interface AD5424/AD5433/AD5445,* One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, USA, 2009.

[11] XILINX Inc., *XA Spartan-6 Automotive FPGA Family Overview, DS170 (v1.3),* December 13, 2012.

[12] Allegro MicroSystems LLC, *Triple Output Step-Down Switching Regulator, 4490-DS, Rev. 6,* 115 Northeast Cutoff, Worcester, Massachusetts USA, 2014.

[13] Maxim Integrated, *MAX1183 Dual 10-Bit, 40Msps, 3V, Low-Power ADC with Internal Reference and Parallel Outputs, Rev 1*, 2006.

[14] Texas Instruments Inc., *VCA2614 Dual, VARIABLE GAIN AMPLIFIER with Input Buffer*, 2001.

[15] XILINX Inc., *UG683, EDK Concepts, Tools, and Techniques*, 2012.

[16] XILINX Inc., *DS643, LogiCORE IP, Multi-Port Memory Controller (v6.06.a)*, February 22, 2013.

[17] XILINX Inc., "MicroBlaze Soft Processor Core," 2014. [Online]. Available: http://www.xilinx.com/tools/microblaze.htm. [Accessed 12 March 2014].

[18] XILINX Inc., *DS865, LogiCORE IP MicroBlaze Micro Controller System (v1.0)*, 2012.

[19] XILINX Inc., *UG388, Spartan-6 FPGA Memory Controller User Guide (v2.3)*, 2010.

[20] XILINX Inc., *UG416, Spartan-6 FPGA Memory Interface Solutions User Guide*, 2011.

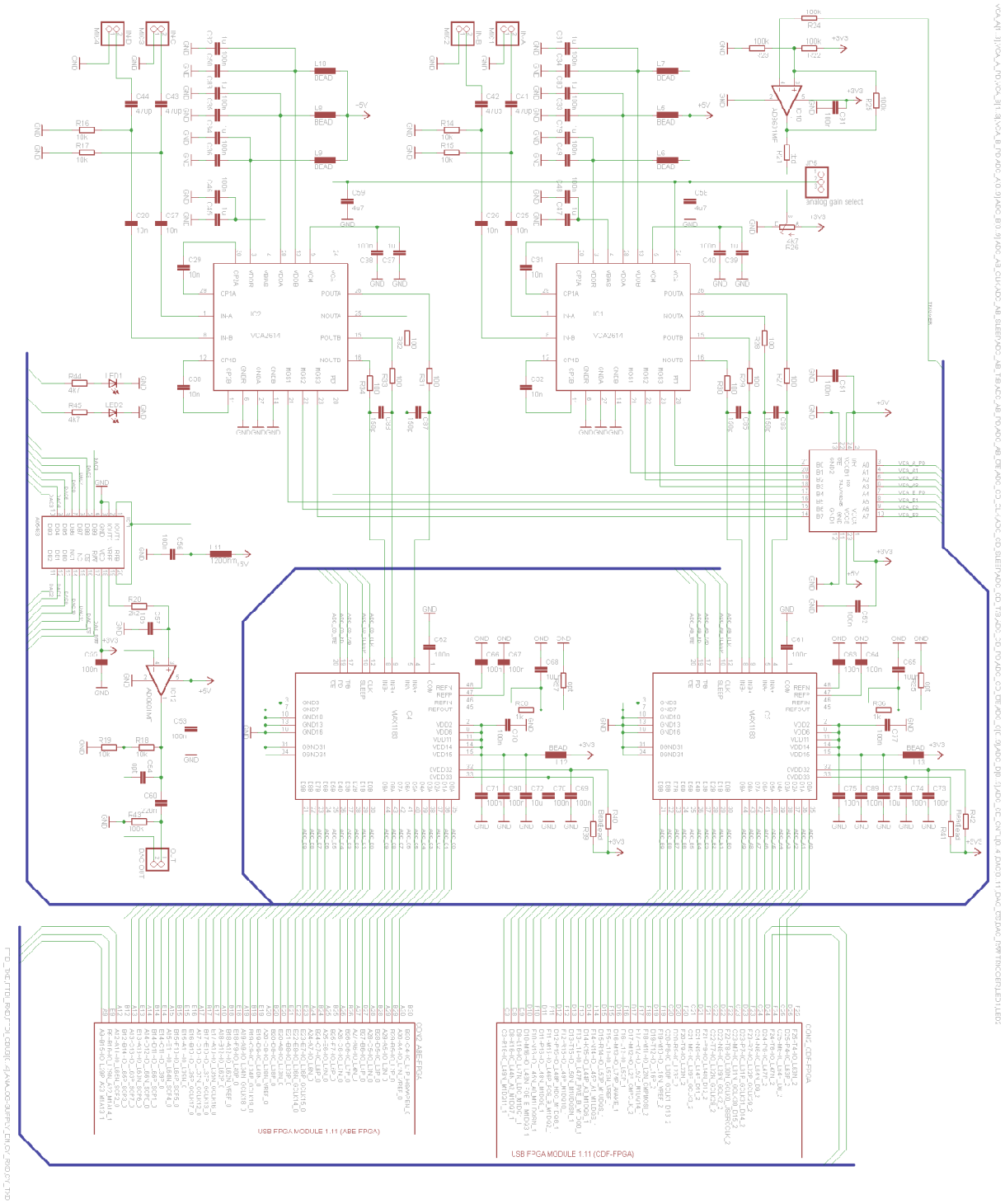
[21] Micron Technology Ltd., *DOUBLE DATA RATE SD RAM 46V32M16*, 2001.

[22] XILINX Inc., *Processor Local Bus (PLB) Arbiter Design Specification*, 2002.

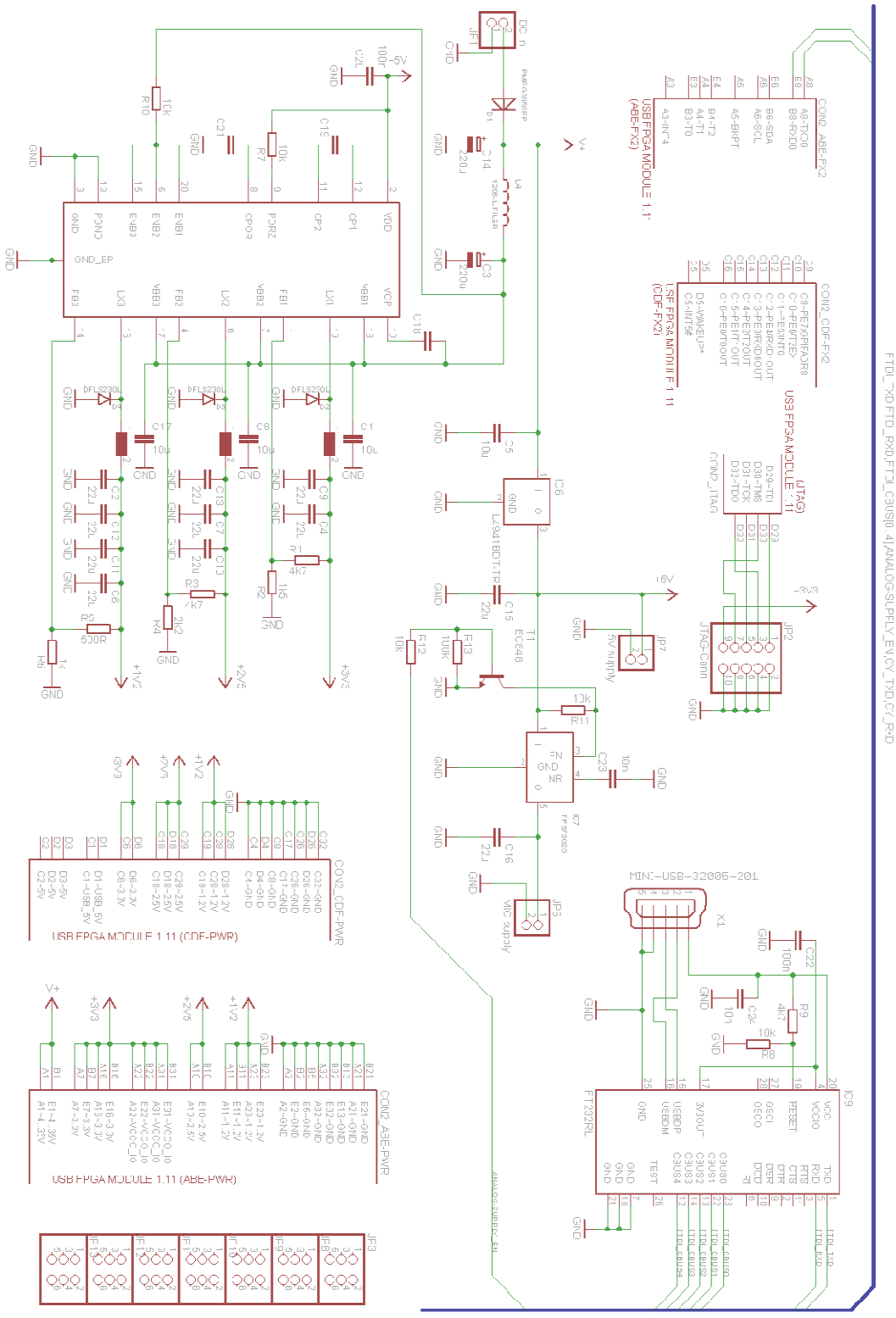
[23] XILINX Inc., *UG382 (v1.9) Spartan-6 FPGA Clocking Resources*, 2013.

Appendix A

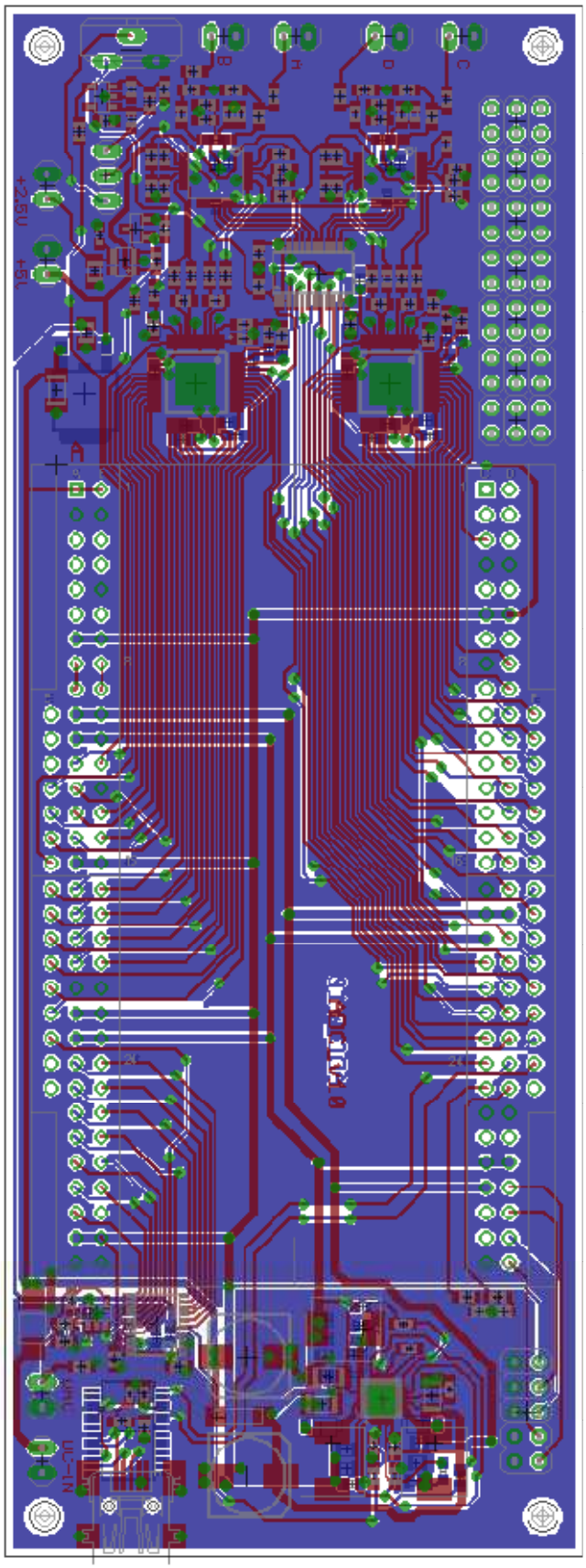
Schematic part I



Schematic part II



Layout



Picture of Prototype

Pin list of ZTEX evaluation board and FPGA

Pin Name (Eagle-Sch.)	FPGA Port	Port Name (XPS)	Function
ADC_AB_CLK	B10~IO_L35P_GCLK17_0	ADC1_CLK	Clock out (custom IP)
ADC_AB_SLEEP	D11~IO_L66P_SCP1_0	Generic_GPIO 0	MicroBlaze GPIO ADC SLEEP
ADC_AB_T/B	A13~IO_L63N_SCP6_0	Generic_GPIO 1	MicroBlaze GPIO ADC T/B
ADC_AB_PD	D12~IO_L66N~SCP0_0	Generic_GPIO 2	MicroBlaze GPIO ADC PD
ADC_AB_OE!	C11~IO_L39P_0	Generic_GPIO 3	MicroBlaze GPIO ADC OE!
ADC_A0	C9~IO_L38P_0	ADC_data 0	ADC Data (custom IP)
ADC_A1	A9~IO_L34N_GCLK18_0	ADC_data 1	ADC Data (custom IP)
ADC_A2	D9~IO_L40N_0	ADC_data 2	ADC Data (custom IP)
ADC_A3	D8~IO_L38P_0	ADC_data 3	ADC Data (custom IP)
ADC_A4	C8~IO_L38N_VREF_0	ADC_data 4	ADC Data (custom IP)
ADC_A5	A8~IO_L33N_0	ADC_data 5	ADC Data (custom IP)
ADC_A6	B8~IO_L33P_0	ADC_data 6	ADC Data (custom IP)
ADC_A7	E8~IO_L36N_GCLK14_0	ADC_data 7	ADC Data (custom IP)
ADC_A8	N9~IO_L14P_D11_2	ADC_data 8	ADC Data (custom IP)
ADC_A9	M9~IO_L29P_GCLK3_2	ADC_data 9	ADC Data (custom IP)
ADC_B0	F9~IO_L40P_0	ADC_data 10	ADC Data (custom IP)
ADC_B1	B12~IO_L62P_0	ADC_data 11	ADC Data (custom IP)
ADC_B2	A12~IO_L62N_VREF_0	ADC_data 12	ADC Data (custom IP)
ADC_B3	A10~IO_L35N_GCLK17_0	ADC_data 13	ADC Data (custom IP)
ADC_B4	C10~IO_L37N_GCLK12_0	ADC_data 14	ADC Data (custom IP)
ADC_B5	E10~IO_L37P_GCLK13_0	ADC_data 15	ADC Data (custom IP)
ADC_B6	C13~IO_L63P_SCP7_0	ADC_data 16	ADC Data (custom IP)
ADC_B7	A11~IO_L39N_0	ADC_data 17	ADC Data (custom IP)
ADC_B8	E11~IO_L64N_SCP4_0	ADC_data 18	ADC Data (custom IP)
ADC_B9	F10~IO_L64P_SCP5_0	ADC_data 19	ADC Data (custom IP)
ADC_CD_CLK	P8~IO_L30P_GCLK1_D13_2	ADC2_CLK	Clock out (custom IP)
ADC_CD_SLEEP	R9~IO_L23P_2	Generic_GPIO 4	MicroBlaze GPIO ADC SLEEP
ADC_CD_T/B	T9~IO_L23N_2	Generic_GPIO 5	MicroBlaze GPIO ADC T/B
ADC_CD_PD	M10~IOL16N_VREF_2	Generic_GPIO 6	MicroBlaze GPIO ADC PD
ADC_CD_OE!	T12~IO_L52N_M1DQ15_1	Generic_GPIO 7	MicroBlaze GPIO ADC OE!
ADC_C0	M11~IOL2N_CMPMOSI_2	ADC_data 20	ADC Data (custom IP)

ADC_C1	R15~IO_L49P_M1DQ10_1	ADC_data 21	ADC Data (custom IP)
ADC_C2	P15~IO_L48P_HDC_M1DQ8	ADC_data 22	ADC Data (custom IP)
ADC_C3	M15~IO_L46P_FCS_B_M1DQ2_1	ADC_data 23	ADC Data (custom IP)
ADC_C4	P16~IO_L48N_M1DQ9_1	ADC_data 24	ADC Data (custom IP)
ADC_C5	N16~IO_L45N_A0_M1LDQSN_1	ADC_data 25	ADC Data (custom IP)
ADC_C6	M16~IO_L46N_FOE_B_M1DQ3_1	ADC_data 26	ADC Data (custom IP)
ADC_C7	L16~IO_L47N_LDC_M1DQ1_1	ADC_data 27	ADC Data (custom IP)
ADC_C8	K16~IO_L44N_A2_M1DQ7_1	ADC_data 28	ADC Data (custom IP)
ADC_C9	R16~IO_L49N_M1DQ11_1	ADC_data 29	ADC Data (custom IP)
ADC_D0	T15~IO_L50N_M1UDQSN_1	ADC_data 30	ADC Data (custom IP)
ADC_D1	L14~IO_L47P_FWE_B_M1DQ0	ADC_data 31	ADC Data (custom IP)
ADC_D2	K15~IO_L44P_A3_M1DQ6_1	ADC_data 32	ADC Data (custom IP)
ADC_D3	N14~IO_L45P_A1_M1LDQS_1	ADC_data 33	ADC Data (custom IP)
ADC_D4	R14~IO_L50P_M1UDQS_1	ADC_data 34	ADC Data (custom IP)
ADC_D5	L13~IO_L53N_VREF_1	ADC_data 35	ADC Data (custom IP)
ADC_D6	M13~IO_L74P_AWAKE_1	ADC_data 36	ADC Data (custom IP)
ADC_D7	L12~IO_L53P_1	ADC_data 37	ADC Data (custom IP)
ADC_D8	M12~IO_L2P_CMPCLK_2	ADC_data 38	ADC Data (custom IP)
ADC_D9	R12~IO_L52P_M1DQ14_1	ADC_data 39	ADC Data (custom IP)
VCA_A_PD	N6~IO_L64N_D9_2	Generic_GPIO 8	MicroBlaze GPIO VCA PD
VCA_A1	R7~IO_L32P_GCLK29_2	Generic_GPIO 9	MicroBlaze GPIO VCA MSG
VCA_A2	P7~IO_L31P_GCLK31_D14_2	Generic_GPIO 10	MicroBlaze GPIO VCA MSG
VCA_A3	M7~IO_L31N_GCLK30_D15_2	Generic_GPIO 11	MicroBlaze GPIO VCA MSG
VCA_B_PD	T8~IO_L30N_GCLK0_USERCCLK_2	Generic_GPIO 12	MicroBlaze GPIO VCA PD
VCA_B1	N8~IO_L29N_GCLK2_2	Generic_GPIO 13	MicroBlaze GPIO VCA MSG
VCA_B2	T7~IO_L32N_GCLK28_2	Generic_GPIO 14	MicroBlaze GPIO VCA MSG
VCA_B3	P9~IO_L14N_D12_2	Generic_GPIO 15	MicroBlaze GPIO VCA MSG
TRIGGER	B14~IO_L65P_SCP3_0	Generic_GPIO 16	MicroBlaze GPIO Trigger
DAC_R/W!	E7~IO_L36P_GCLK15_0	Generic_GPIO 17	MicroBlaze GPIO DAC R/W!
DAC_CS!	B6~IO_L5P_0	DAC_CLK	MicroBlaze GPIO DAC CS!
DAC0	A5~IO_L2N_0	DAC_data 0	DAC Data (custom IP)
DAC1	B5~IO_L2P_0	DAC_data 1	DAC Data (custom IP)
DAC2	A4~IO_L1N_VREF_0	DAC_data 2	DAC Data (custom IP)
DAC3	D6~IO_L7P_0	DAC_data 3	DAC Data (custom IP)

DAC4	C6~IO_L7N_0	DAC_data 4	DAC Data (custom IP)
DAC5	F7~IO_L5P_0	DAC_data 5	DAC Data (custom IP)
DAC6	E6~IO_L5N_0	DAC_data 6	DAC Data (custom IP)
DAC7	C7~IO_L6P_0	DAC_data 7	DAC Data (custom IP)
DAC8	A7~IO_L6N_0	DAC_data 8	DAC Data (custom IP)
DAC9	A6~IO_L4N_0	DAC_data 9	DAC Data (custom IP)
DAC10	D5~IO_L3P_0	DAC_data 10	DAC Data (custom IP)
DAC11	C5~IO_L3N_0	DAC_data 11	DAC Data (custom IP)
ANALOG-Supply_EN	A14~IO_L65N_SCP2_0	Generic_GPIO 18	MicroBlaze GPIO Supply EN
LED1	M6~IO_L46P_D8_2	fpga_0_LEDS_GPIO_IO_O_pin 0	MicroBlaze GPIO LED1
LED2	P4~IO_L63P_2	fpga_0_LEDS_GPIO_IO_O_pin 1	MicroBlaze GPIO LED2
FTDI_TXD	P6~IO_L47P_2	fpga_0_RS232_TX_pin	MicroBlaze UART
FTDI_RXD	T6~IO_L47N_2	fpga_0_RS232_RX_pin	MicroBlaze UART