DIPLOMA THESIS

Design of a Demonstrator Tool as well as a measurement hardware for characterization and representation of the Infineon current sensor which is based on the Linear Triple Hall principle

Carried out at the Institute of electronic

of the Graz University of Technology

Under guidance of

Prof. Dr. Gerhard Stöckler

In cooperation with

Under guidance of

Dipl.Ing. Mario Motz

&

Dipl.Ing Christian Reidl

By

Theodor Kranz

0430114

Villach, Feb 2011

1 Acknowledgment

First, I thank my supervisors DI Mario Motz and DI Christian Reidl from Infineon Technologies for their helpful experience and support during my work. I also would like to thank Prof. Dr. Gerhard Stöckler from the Institute for electronic of the TU Graz for his readiness and help.

Furthermore I thank my college Michael Strasser for supporting me by developing the demonstration tool as well as for his encouragement.

Additional I thank all my colleagues from the IFAT DC ATV SC D VI department for giving me the possibility to write this diploma thesis.

Last but not least I would like to thank Joris van Rantwijk for his online support from the Netherlands. Thank you for providing me the RTL-code to implement the USB-communication on the FPGA and for helping me to adapt the code for my thesis.

Theodor Kranz

Design of a Demonstrator Tool as well as a measurement hardware for characterization and representation of the Infineon current sensor which is based on the Linear Triple Hall principle

Measuring current with a very high accuracy takes a great part in the field of energy efficiency applications. Various processes with and without galvanic isolation are used. In this thesis a demonstration tool based on FPGA has to be developed in order to evaluate the new Infineon current sensor based on the linear Hall Effect. This hardware allows demonstrating the functionality of the sensor. The innovative measure principle of the current sensor enables to measure current with a high accuracy in a wide range. Thereby the magnetic field of the current will be measured. This has the advantage of an overload proof measure principle because of the galvanic isolation. Due to the used Triple Hall principle the rejection of an interfering field can be detected and canceled from the wanted signal. The current senor is based on present linear Hall sensors from Infineon and delivers a digital signal at each of the three channels. The output signals are containing the magnetic field information as well as temperature and stress information. Due to spreads which occur in semiconductor production, the output signal of the sensor is dependent on the influence of stress and temperature. Therefore the chip has implemented analog compensation techniques to pre-compensate these impacts. For accurate measurements, the residual error has to be compensated at the output of the sensor. The digital data processing of the magnetic field, temperature and stress signals are the focus of this thesis. The current sensor combined with the FPGA based hardware represents a current sensing demonstration system. Measurements and analyzes of the sensor by using the developed hardware tool allows to recommend compensation algorithm for the current sensor.

Design of a Demonstrator Tool as well as a measurement hardware for characterization and representation of the Infineon current sensor which is based on the Linear Triple Hall principle

Für Applikationen im Bereich der Energieeffizienz von Systemen stellt das hochgenaue Messen von Strömen einen wesentlichen Bestandteil dar. Dazu kommen unterschiedlichste Verfahren mit und ohne galvanische Kopplung zum Einsatz. Ziel dieser Diplomarbeit ist es einen Demonstrator auf FPGA Basis zu erstellen um damit die Evaluierung eines auf Basis des Halleffekt arbeitenden neuartigen Strommessprinzips durchzuführen, bzw. für Kunden eine Plattform für die Evaluierung von Algorithmen zur Verfügung zu stellen. Der zum Einsatz kommende Stromsensor arbeitet auf einen Triple Hall Prinzip und ermöglicht eine galvanisch entkoppelte, Störfelder unabhängige, Überlast geschützte, mit einem großen Dynamikumfang durchführbare Strommessung. Die vorliegenden Stromsensor Bauelemente sind aktuell auf Basis von vorhandenen Linearen Hallsensoren aufgebaut und liefern pro Kanal ein digitales Signal, das jeweils Temperatur, Stress und Magnetfeldinformationen enthält. Die Verarbeitung dieser digitalen Signale ist Ziel der Diplomarbeit. Weiteres soll dann in Folge für einen speziellen Test-Chip eine FPGA Lösung erarbeitet werden die dann ein vollständiges Stromsensorsystem nachbildet und für die Evaluierung von Auswertealgorithmen dienen kann.

3 Table of contents

4 Tasks and objectives

This document describes the design of a measurement tool which determines the characteristics of the current sensor. Additionally, this tool also operates as a demonstrator tool showing the functionality of the sensor. The current sensor has been developed as a part of an innovation project and allows measuring the magnetic field. The current which produces the magnetic field can be determined in a high dynamic range. Therefore the current can be measured with the advantage of a galvanic isolation. Interfering fields are rejected by using the Triple Hall Effect. Thus interference fields are detected and removed from the desired signal. The sensor consists of derivatives of the linear Hall Sensor TLE4998. To guarantee stable signals, the chip uses compensation techniques before the signal is converted in the digital domain by the analog to digital converters. This compensation techniques like chopping, stress compensation, technology spread compensation or linear temperature coefficient compensations ensure stable and accurate signals. The main focus by designing the chip was to guarantee a stable and accuracy performance over lifetime. Due to the exact reproducibility of the measure signals, the residual sensitivity and offset error of the Hall sensor can be compensated by digital filters and data processing at the output of the chip. The current sensor delivers a digital signal for each of the three channels, which contains the temperature, mechanical stress and magnetic field information. The digital processing of this data, based on a FPGA is the focus of this diploma thesis. The representation of the current in amperes as well as the functionality of the interference field rejection is done with a PC. Thus an interface between the PC and the signal process unit is essential. Before the data from the current sensor can be processed by the FPGA the current sensor has to be set to a stable operating state. Therefore a user interface is necessary to setup the chip. Furthermore a special circuit is required to modulate the supply voltage of the sensor interface, which allows setting the sensor in the test mode to operate the chip in various operating modes.

5 Current and magnetic field

5.1 Current

The electrical current is defined as the flow of charges. Therefore the change of the charge has to be analyzed in respect to the time. This is described in Equation 5.1 where the charge is differentiated in respect to the time.

$$I := \frac{dQ}{dt}$$

Equation 5.1 [14]

The unit of the defined current is given in ampere.

The current density j can be described as the current which crosses in a perpendicular way through a defined area. Equation 5.2 represents the current by integrating the current density over the area.

$$I = \int_A \vec{j} d\vec{A}$$

Equation 5.2 [1]

Dependent on the material of the conductor, the charge carrier are mainly electrons and ions. The amount of the current is dependent on the charges velocity as well as on their amount of charges. This is described in Equation 5.3.

$$I = n \cdot q \cdot \vec{A} \cdot \vec{v}$$

Equation 5.3 [1]

Equation 5.4 is used by considering a current flow through a closed surface.

$$I = \oint_A \vec{j} d\vec{A} = -\frac{dQ}{dt} = -\frac{d}{dt}\int \rho dV$$

Equation 5.4 [1]

By using the gauss' law to convert a surface integral into a volume integral Equation 5.4 can be written as

$$\oint_A \vec{j} d\vec{A} = \int_V div\vec{j} dV$$

Equation 5.5 [1]

Out of Equation 5.5 the Equation 5.6 can be derived, which is also known as the continuity equation.

$$div\vec{j} = -\frac{\partial\rho}{\partial t}$$

Equation 5.6 [1]

If the charge density ρ is time invariant the Equation 5.6 and Equation 5.7 consider, that the current density is solenoidal.

$$I = \oint_A \vec{j} d\vec{A} = 0 \rightarrow div\vec{j} = 0$$

Equation 5.7 [1]

5.2 Correlation of the magnetic and electrical field

The interrelationship between the magnetic field and the current, respectively the electric field can be described by the Maxwell equations. The equations are describing how a magnetic field will be generated by a current, respectively by a time variant electrical field. Also the inverse effect, generating an electrical field by a time variant magnetic field, can be explained by the Maxwell equations.

For the derivations of the Maxwell equations the divergence theorem, also known as the Gauss' theorem is used. The Stokes theorem which is known as the curl theorem is also a useful calculation specification, to derive the Maxwell equations.

Equation 5.8 describes the Gauss 'theorem

$$\iiint_V div \cdot \vec{F} \cdot dV = \oiint_S \vec{F} \cdot d\vec{A}$$

Equation 5.8 [4]

The Gauss theorem is used to transform a volume integral into a surface integral across the external side of the lateral surface. [15]

Equation 5.9 describes the circulation theorem

$$\oint_C \vec{F} \cdot d\vec{s} = \int_S rot\vec{F} \cdot d\vec{A}$$

Equation 5.9 [22]

The electrical flux is described in Equation 5.10.

The electrical field through a defined area is described as the electrical flux. If the relation will be executed to a closed surface, which includes a charge Q the total electrical flux is given by Equation 5.10.

$$\phi_{el} = \oint_A \vec{E} d\vec{A}$$

Equation 5.10 [1]

By using the Gauss' theorem Equation 5.10 can be written as shown in Equation 5.11.

$$\phi_{el} = \oint_A \vec{E} d\vec{A} = \int_{v(a)} div \vec{E} dV$$

Equation 5.11 [1]

By substituting the *E* by the point charge, Equation 5.11 is written as follows.

$$\phi_{el} = \frac{1}{\varepsilon_0} Q = \frac{1}{\varepsilon_0} \int_V \rho dV$$

Equation 5.12 [1]

In differential form Equation 5.12 is written as

$$divE = \frac{1}{\varepsilon_0} \rho$$

Equation 5.13 [1]

Equation 5.13 shows that the sources of the electrical field are the positive charges. This formula is the third equation of the Maxwell equations and also known as the gauss law. [1]

The gauss' law states that the electrical flux through a surface, which includes a charge, will be proportional to the total electric charge enclosed by the surface.

The magnetic field intensity is the analog part to the electric field strength. And is defined in Equation 5.14

$$\vec{B} = \frac{\vec{F}}{p}$$

Equation 5.14 [1]

F stands for the force produced by the magnetic field causing on the magnetic pole *p*.

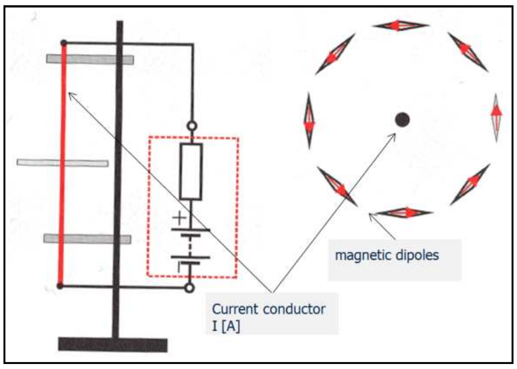


Figure 5.1 : Setup to determine the magnetic field produced by a current flow in a straight current conductor [1]

The experiment shown in Figure 5.1 represents that the magnetic dipoles around a straight wire in an electric circuit are straightening them in the direction of the magnetic field. The dependence of the magnetic field from the amperage, length and distance r is described in Equation 5.15.

$$B_{(r)} = \frac{\mu_0 I}{2\pi r}$$

Equation 5.15 [16]

The direction of *B* can be determined by the right-hand rule. If the thumb indicates in the direction of the technical current flow, the fingers are giving the direction of the magnetic field.

The curl of the magnetic field can be described in a mathematical way.

$$\oint_C \vec{B} d\vec{r} = \frac{\mu_0 I 2\pi r}{2\pi r} = \mu_0 I$$

Equation 5.16 [1]

Equation 5.16 shows that the path of integration is independent to the result. The evidence therefore can be given with the Gauss 'law. Equation 5.16 is also known as the Ampers' law. This formula is used in Biot-Savarts' Law to calculate the magnetic field from a piece of a current conductor. Together with the Lorentz force the attractive force, causing on a charge by an electron in a wire is used to describe the relativistic correlation between the electrical and the magnetic fields.

By using the Stokes theorem and Equation 5.7, Equation 5.16 is written in a differential form.

$$rot\vec{B} = \mu_0 \vec{j}$$

Equation 5.17

The magnetic flux is described in Equation 5.18

$$\phi_m = \int_A \vec{B} d\vec{A}$$

Equation 5.18 [4]

Because of the rotation of the magnet field, the flux through a surface s is zero.

By using the Gauss law, Equation 5.19

$$\oint_S \vec{B} d\vec{S} = 0$$

Equation 5.19 [1]

can be written as shown in Equation 5.20.

$$div\vec{B} = 0$$

Equation 5.20 [1]

Equation 5.20 indicates that no magnetic monopole can exist. Therefore the magnetic field of an electrical current is a solenoidal curl field. Equation 5.20 is known as the fourth Maxwell equation.

The two mentioned Maxwell equations are describing the behavior of the stationary electromagnetic fields. By considering the law of induction which is represented in Equation 5.21 and the displacement-current from Equation 5.22 the Maxwell equations for time dependent fields can be derived.

$$U_{ind} = -\frac{d\phi_m}{dt} = \frac{-d}{dt}\int_A \vec{B}d\vec{A}$$

Equation 5.21 [14]

$$I_d = \varepsilon_0 \frac{d}{dt}\int \vec{E}d\vec{A}$$

Equation 5.22

If the variation of the magnetic field only depends to the time, Equation 5.21 can be written as follows

$$U_{ind} = -\int \frac{\partial \vec{B}}{\partial t} d\vec{A}$$

Equation 5.23

By considering that the voltage is calculated by integrating the electrical field with a line integral as shown in Equation 5.24.[1],

$$U = \oint_C \vec{E}d\vec{r}$$

Equation 5.24 [1]

the law of induction can be written as follows

$$\oint_C \vec{E}d\vec{r} = -\frac{d}{dt}\int_A \vec{B}d\vec{A}$$

Equation 5.25 [1]

Equation 5.25 can be written in the differential form as shown in Equation 5.26

$$rot\vec{E} = -\frac{\partial \vec{B}}{\partial t}$$

Equation 5.26 [5]

The Equation 5.26 is known as the first of the four Maxwell equations.

Equation 5.26 shows that a time variant magnetic field causes an electrical curl field.

The Amperes law extended with the dielectric current is described in Equation 5.27 and represents the second Maxwell equation.

$$\oint_C \vec{B} d\vec{r} = \mu_0 I + \frac{1}{c^2}\frac{d}{dt}\int \vec{E} d\vec{A}$$

Equation 5.27 [1]

$$rot\vec{B} = \mu_0 \vec{j} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t}$$

Equation 5.28 [5]

Equation 5.28 represents the second Maxwell equation in a differential form and specifies that a current in a wire as well as the dielectric current, for example the electrical field in a capacitor, causes a curl magnetic field around the current.

The correlation between the magnetic field and the electrical field can be described with the theory of relativity (A. Einstein) and the Lorentz-Transformation. Thereby the forces between two parallel moving charges with the distance r and the velocity *v*, in a static coordinate system and in a moving coordinate system will be compared. [22]

6 Lorentz force

One important fact, which is demonstrated in the experiment shown in Figure 6.1, is that a defined magnetic field *B* causes a force *F* of a charge *q* which has a defined velocity *v*. The force is always normal to the velocity vector and also normal to the vector of the magnetic field.

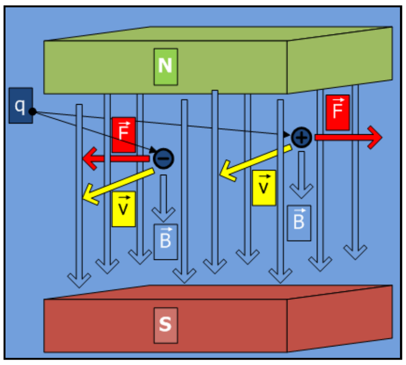


Figure 6.1: Charges in magnetic field

The force in Figure 6.1 can be calculated as shown in Equation 6.1 and is called the Lorentz force.

$$\vec{F} = q(\vec{v} \times \vec{B})$$

Equation 6.1 [16]

If an electrical field causes auxiliary to the charge *q* the force is calculated as shown in Equation 6.2.

$$\vec{F}_{all-up} = q(\vec{E} + \vec{v} \times \vec{B})$$

Equation 6.2 [16]

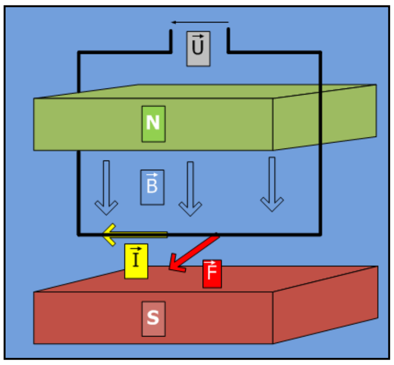


Figure 6.2 : current conductor in a magnetic field

Figure 6.1 demonstrates the force on a charge in motion. The experiment in Figure 6.2 describes, that the current flow, respectively the flow of the electrons in a straight wire inside a magnetic field causes a force, which affects the wire out from the magnetic field in a perpendicular way. The force is the same as described in Figure 6.1 but the electrons are moving on a given way dependent on the current conductor. The force which affects to the wire is called the Lorentz force.

In assumption that the current through the wire is calculated as described in Equation 6.3, the force can be calculated as described in Equation 6.4.

$$I = n \cdot q \cdot v \cdot A$$

Equation 6.3

$$dF = n \cdot A \cdot dL \cdot q \cdot (\vec{v} \times \vec{B}) = (\vec{j} \times \vec{B})dV$$

Equation 6.4

$$\vec{F} = \int_{L_1}^{L_2} (\vec{j} \times \vec{B})dV$$

Equation 6.5 [22]

The Lorentz force which causes to the wire in Figure 6.2 can also be written as shown in Equation 6.6.

$$\vec{F} = l \cdot \vec{I} \times \vec{B}$$

Equation 6.6 [4]

Because of the fact of Equation 6.7

$$\vec{I}l = \vec{j}Al = \frac{N}{V}e\vec{v}V = Q\vec{v}$$

Equation 6.7 [1]

the Lorentz force can also be written as shown in Equation 6.8.

$$\vec{F} = Q\vec{v} \times \vec{B}$$

Equation 6.8 [1]

Equation 6.6 is used to determine the unit of the magnetic field by the premise that the current is measured in ampere.

$$[B] = 1\frac{N}{Am} = 1\frac{Vs}{m^2} = 1T$$

Equation 6.9 [1]

7 Hall Effect

As mentioned, the Lorentz force takes effect to the charge carrier within a magnetic field. Thus the electrons inside a conductor with an expanded transversal section do not move on a straight direction. The charges are deflected by the Lorentz force due to the magnetic field in a perpendicular way to the current direction. This effect can be seen in Figure 7.1.

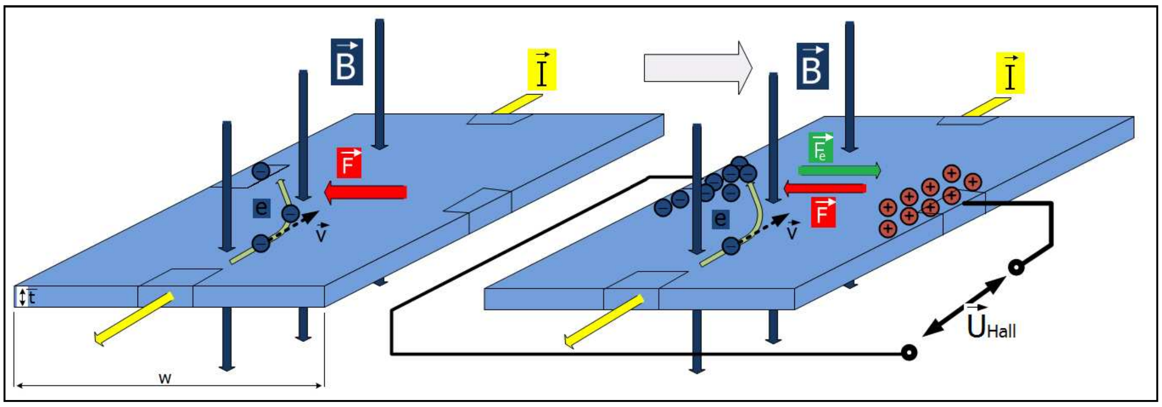


Figure 7.1 : Hall Effect

The left side of Figure 7.1 shows a Hall plate and the direction of the supply current flow. The yellow arrow indicates the technical current flow. To describe the Hall Effect by the Lorentz force the physical current direction has to be examined. In Figure 7.1 the direction of the electrons with the velocity *v* are represented by the black marker. Due to the magnetic field, the electrons are deflected in a normal direction to the velocity vector and the magnetic field vector as shown in Figure 7.1. Due to the change of the electron concentration an electrical field which is counteracting to the Lorentz force is the result as indicated in the right side of Figure 7.1. If the two forces are in equilibrium, the potential difference, dependent on the electrical field, at the outer face of the Hall plate is representing the Hall voltage U_{Hall}. If the movement of the charges to the lateral side of the Hall plate stops due to the balance of the forces, the Hall voltage can be calculated as described below.

As mentioned the Lorentz force and the electrical field force can be equalized as shown in Equation 7.1.

$$F_{Lorentz} = F_{e-field} = Ee = evB\big|_{\angle 90^\circ}$$

Equation 7.1 [4]

And *E* can also be written as follows.

$$E = U_{Hall} / w$$

Equation 7.2 [4]

Therefore the voltage U_{Hall} is calculated as shown in Equation 7.3.

$$U_{Hall} = wvB$$

Equation 7.3 [4]

The velocity *v* of the electrons can be substituted by the current *I*.

It is essential that

$$v = \frac{l}{t}$$

Equation 7.4

$$l = \frac{V}{A} = \frac{V}{w\bar{t}}$$

Equation 7.5 [4]

$$n = \frac{N}{V} \Rightarrow V = \frac{N}{n} = \frac{Q}{ne}$$

Equation 7.6 [4]

$$Q = It$$

Equation 7.7 [4]

Therefore the velocity can be shown as in Equation 7.8.

$$v = \frac{I}{b\bar{t}ne}$$

Equation 7.8 [4]

By substituting the *v* in Equation 7.3 the Hall voltage can be calculated as shown in Equation 7.9.

$$U_{Hall} = \frac{IB}{ne\bar{t}} = RH\frac{IB}{\bar{t}}$$

Equation 7.9 [4]

The reciprocal value of the Hall coefficient *RH* describes the charge density. [4]

8 Filter

8.1 Decimation filter

Hogenauer-CIC-filter

The cascaded integrator–comb filter is an efficient structure to perform decimation and also interpolation filters. This kind of filter structure has the benefit that it works without multipliers, which require a lot of operating time and memory from the system. So it is possible to reduce the computational complexity for narrowband low pass filters. This FIR filter operates on reduced clock rates and thus guarantees minimization of power consumption in high speed hardware. This is one of the advantages that make this filter an economical alternative to conventional filter implementations. The comb filter is also optimal for use with hardware implementations like FPGAs. The principal function of the decimation and the interpolation filter respectively is to decrease or increase the sampling rate and to keep the pass band aliasing or imaging error within defined bounds. [10]

The filter structure consists of cascaded ideal integrator stages which operate at high sampling rate. Thereby the intermediate storage range is reduced due to the integration at high sampling rate and the comb filtering at a lower sampling rate. The very "regular" structure of the CIC filter makes it easy to program the filter and to implement it in hardware. There has to be an equal number of comb stages, which are operating at a low sampling rate of the system, to the number of integrators. Together, a single-integrator comb pair produces a uniform FIR. [10] Another advantage of this filter structure is the flexibility of the sampling rate. The CIC filter can be designed easily with a programmable rate change. The internal register length of the filter structure depends on the length of the input and output registers and from its down sampling factor. Therefore, the structure of the filter need not be changed by varying the down sampling rate of the system. Equation 8.6 describes how the register length is calculated. The register length has to be changed to the corresponding down sampling value if the decimation factor is changed.

8.1.1 Building Blocks

The CIC filter consists of two basic building blocks with a rate change switch between the high and low sampling stages. One block is the integrator, which is also known as an accumulator. The integrator operates at the high sampling rate. The second part of the filter is the comb stage which consists of a differentiator, operating at the low sampling rate. [12] The software chapter describes the adjusted sampling rates. Depending on the update rate from the sensor, the higher sampling rate works at one MHz. Due to the down sampling factor, the lower sampling rate works at four kHz. The integrator section consists of a number of ideal digital integrator stages, which operate at the high sampling rate. Each of these stages is implemented as a one-pole filter which has a unity feedback coefficient. Figure 8.1 shows a basic integrator.

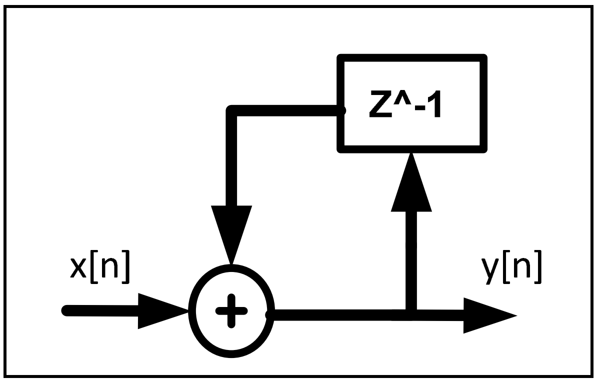


Figure 8.1 : Basic Integrator

Equation 8.1 describes the difference equation of the integrator.

$$y[n] = y[n-1] + x[n]$$

Equation 8.1 [13]

The single integrator transfer function in the z–domain is described by Equation 8.2.

$$H_I(z) = \frac{1}{1 - z^{-1}}$$

Equation 8.2 [12]

The comb section operates at the low sampling rate *f_s/R*. *R* describes the change factor of the sampling rate. In the case where the comb filter is used as an interpolation filter, there will be $R-1$ zeros value samples inserted between the samples of the comb section output. Figure 8.2 shows a basic differentiator. [13]

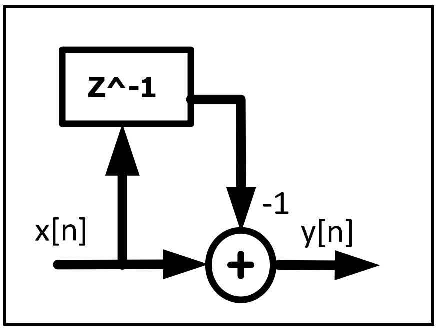


Figure 8.2 : Basic Differentiator

Equation 8.3 describes the difference equation of the comb filter which describes the structure of a FIR filter.

$$y[n] = x[n] - x[n-1]$$

Equation 8.3 [13]

The transfer function of the system in the z domain is described by Equation 8.4.

$$H_C(z) = 1 - z^{-RM}$$

Equation 8.4

The design parameter M is called the differential delay. This is a filter design parameter which is used to control the filter's frequency response. This parameter is usually held to one or two in practice. For the decimation, the rate change-switch subsamples the output of the last integrator stage by reducing the sampling rate from f_s to f_s/R. Figure 8.3 shows the structure of the CIC filter with the down sampling switch, which reduces the sampling rate.

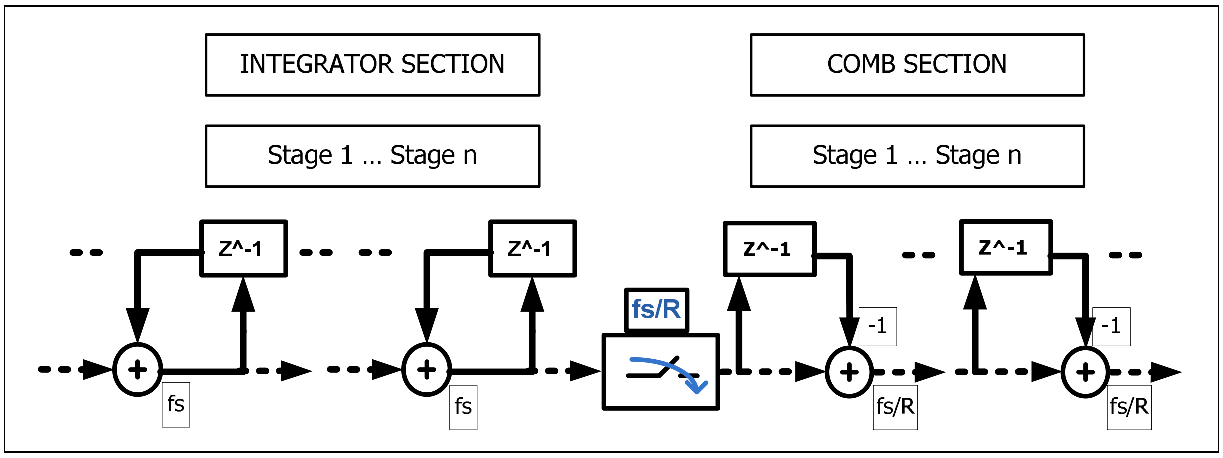


Figure 8.3 : Structure of the CIC filter

Due to the interconnection of the integrator section with the comb section the overall system function is calculated as follows.

$$H(z) = H_I^{\ N}(z) \cdot H_C^{\ N}(z) = \frac{(1 - z^{-RM})^N}{(1 - z^{-1})^N} = \left[\sum_{k=0}^{RM-1} z^{-k} \right]^N$$

Equation 8.5 [13]

Figure 10.27 shows the structure of the used comb filter.

The required bit length of the internal register of the CIC filter has to be calculated with the formula Equation 8.6.

$$B_{\max} = N \cdot \log_2(RM) + B_{in} - 1$$

Equation 8.6 [10]

Due to the pole at the unit cycle from the integrator transfer function the IIR filter is not stable. Register overflow in the integrator stages could be possible. This is without negative consequences if the following two constraints are met. First the arithmetic inside the filter structure has been implemented in a format which allows "wrap-around" between the most positive and the most negative numbers as the two's complement is. The second condition requires the range of the number system having the same amount or exceeds the maximum magnitude expected at the output of the composite filter output. For CIC interpolators the data is preconditioned at the differentiator stage to avoid overflow occurring in the integrator section. [10]

At the input stage of the filter an integrate dump algorithm has been used to reduce the number of adders at this stage. Therefore the input values will be added *k* times consecutively. After this value will be transferred to the output a reset occurs. *K* corresponds to the order of the filter.

8.1.2 Frequency characteristic

By changing only the three programmable values 'order N', 'down sampling factor R' and the 'differential delay M', the frequency response can be determined exactly. [10]

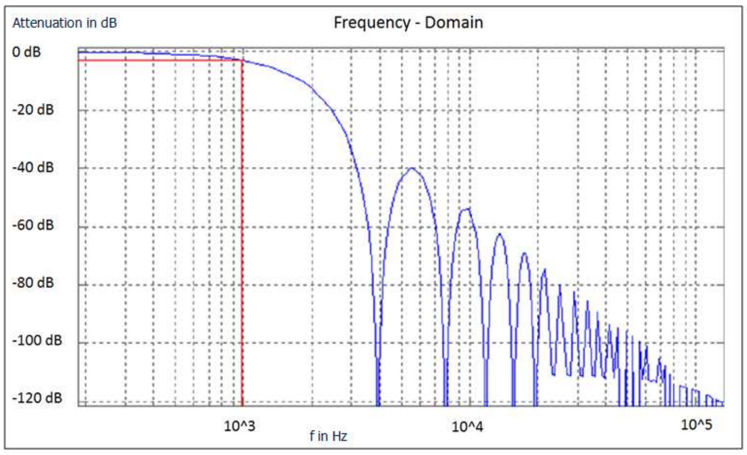


Figure 8.4 : Frequency domain of the decimation filter

Figure 8.4 shows the frequency response for N=3, M=1, R=256. These settings have also been used in the implemented filter structure on the FPGA. Figure 8.4 shows the frequency response of the filter where f is the frequency relative to the low sampling rate f_s / R . The cut off frequency of the system is at 1000 kHz. The power response is described by Equation 8.7.

$$P_{(f)} = \left[\frac{\sin \cdot \pi \cdot M \cdot f}{\sin \dfrac{\pi \cdot f}{R}} \right]^{2N}$$

Equation 8.7 [10]

This power response can be approximated over a defined frequency range, if the rate change factor R is set high enough, by Equation 8.8.

$$\hat{P}(f) = \left[RM \frac{\sin(\pi f M)}{\pi f M} \right]^{2N} \bigg| for \to 0 \le f < \frac{1}{M}$$

Equation 8.8 [10]

Zeros occur in the power response at multiples of $f = 1/M$. The differential delay factor M can be used to set the placements of zeros.
Due to the decimation a spectral folding at the CIC filter takes place. The desired band width B which is shown in Figure 8.5 highlighted in green is centered at multiplies of f_s/R. The aliased energy after the decimation depends on the filter order and the bandwidth range. The smaller the bandwidth, the lower the energy of aliasing after the decimation.[underst] Table 8.1 represents the aliasing attenuation depending on the differential delay M and the relative bandwidth f_c.

The aliasing/imaging bands are as follows.

$$(i - f_C \le f \le (i + f_C))$$

Equation 8.9 [10]

Figure 8.5 shows an example of the frequency response for *N*=4, *M*=1, *R*=7 and f_c = 1/8. The pass band cutoff is at $f_c = 1/8$. The aliasing is centered on zero at the frequency values one, two and three, which are relative to the down sampling frequency.

Differential Delay M	Relative Bandwidth f_C	Aliasing/Imaging Attenuation in dB As a Function of Number of Stages (N)					
		1	**2**	**3**	**4**	**5**	**6**
1	**1/128**	42.1	84.2	126.2	168.3	210.4	252.5
1	**1/64**	36	72	108.3	144	180	215.9
1	**1/32**	29.8	59.7	89.5	119.4	149.2	179
1	**1/16**	23.6	47.2	70.7	94.3	117.9	141.5
1	**1/8**	17.1	34.3	51.4	68.5	85.6	102.8
1	**¼**	10.5	20.9	31.4	41.8	52.3	62.7
2	**1/256**	48.1	96.3	144.6	192.5	240.7	288.8
2	**1/128**	42.1	84.2	126.2	168.3	210.4	252.5
2	**1/64**	36	72	108	144	180	216
2	**1/32**	29.9	59.8	89.6	119.5	149.4	179.3
2	**1/16**	23.7	47.5	71.2	95.0	118.7	142.5
2	**1/8**	17.8	35.6	53.4	71.3	89.1	106.9

Table 8.1 : Attenuation in dB as a function of stages, band width and differential delay [10]

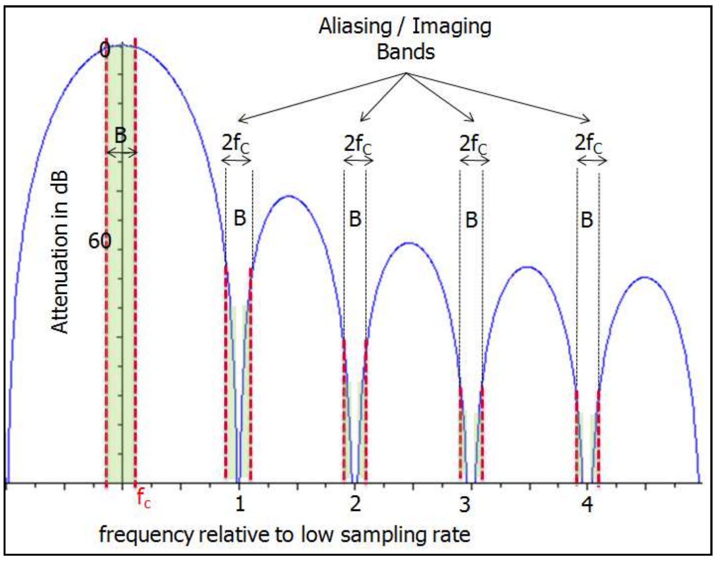


Figure 8.5 : frequency response *N*=4, *M*=1, *R*=7, f_c=1/8

8.2 IIR

The IIR filter has its name due to the infinite impulse response given by the filter structure. The filter structure can be described with signal flow charts as shown in Figure 8.6. The difference equation which describes the filter structure can be derived from the signal flow chart. Equation 8.10 represents the difference equation of the chart in Figure 8.6-a.

$$y_{[n]} = ay_{[n-1]} + x_{[n]}$$

Equation 8.10 [2]

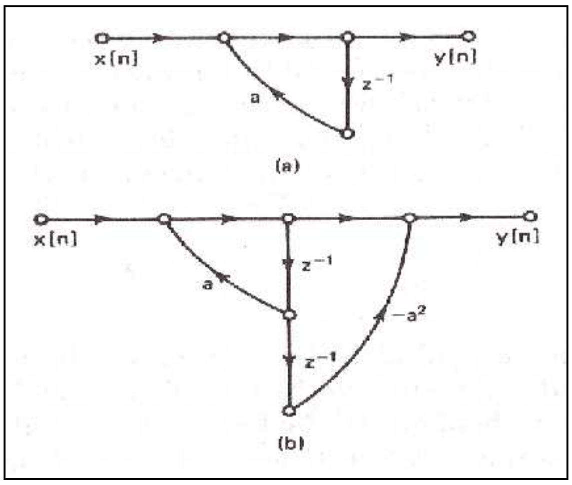


Figure 8.6 : flow chart with feedback loop [2]

A feedback loop as shown in Figure 8.6-a are necessary but not sufficient to produce an infinite impulse response. That can be shown by examining a structure without any loops. In such a case, the maximum delay of a signal from the input to the output cannot excess the maximum number of delay stages in the system. Thereby the impulse response for a system without any feedback loops cannot be greater than the number of the delay stages in a system. Thus the number of zero points in a system function of a loop less system cannot be greater than the number of the delay stages in the system. By considering the impulse response of the system from Figure 8.6-a depending on the factor a, the amplitude at the output will either grow or decrease. The impulse response is given by Equation 8.11. [2]

$$h_{[n]} = a^n u_{[n]}$$

Equation 8.11 [2]

Thereby a feedback loop can produce an infinite impulse response. If a system function contains poles, then the corresponding flow chart will consist of feedback loops. But neither poles nor feedback loops are sufficient for infinite impulse responses. Figure 8.6-b shows a structure with loops, but the impulse response of this system is finite. Because the poles of the system function, which describes Figure 8.6-b consist of poles which cancel the zeros. This system is described by Equation 8.12.

$$H(z) = \frac{1 - a^2 z^{-2}}{1 - az^{-1}} = \frac{(1 - az^{-1})(1 + az^{-1})}{1 - az^{-1}} = 1 + az^{-1}$$

Equation 8.12 [2]

The impulse response of this system is given as

$$x[n] = \partial[n] + a\partial[n-1]$$

Equation 8.13 [2]

This is a simple example for a fir-system.

The z-transfer functions of IIR filter consist of adjustable poles and zero points. Analog filters consist also of poles and zeros. Thereby IIR filters can be used to simulate analog filters by adjusting the zeros and poles as required. Therefore the pole, respectively zero schemas has to be transformed from the s domain to the z domain. That means that digital filters can be designed by the model of a known analog filter like Butterworth, Tschebycheff or Cauer – filter for example. [6]

The IIR filters are suitable to design filters in the digital domain to implement them in hardware like FPGA or ASICs.

The frequency transformation which has been done to map the specifications from the analog part to the digital domain can be done by the impulse invariant transformation or the bilinear transformation for example. The bilinear transformation is transforming the total transfer function into the analog domain and therefore no aliasing occurs. The impulse invariant transformation is an approximation of the BLT. In this thesis only the BLT is described.

8.3 BLT

IIR design by bilinear transformation

To design a filter in the digital domain to implement it into a PC, a known filter function from the analog domain can be transformed. Therefore different transformations, which are known as IIT or BLT, can be used. As mentioned, no aliasing occurs by using the BLT. Another advantage of this transformation is that it describes a stable transformation. In other words, a stable transfer function in the analog domain leads to a stable transfer function in the digital domain after the bilinear transformation has been done. In this thesis a low pass filter has been implemented in the hardware. Therefore an IIR filter has been designed to implement the low pass in RTL code. A given analog transfer function which has to be transformed into the digital domain consist of adders, multipliers and integrators. The adder and multiplier structure can be adapted in digital without any transformation. Dependent on the register length, multiplication can be done on the hardware as well as additions. The integration in the analog part is done by operation amplifiers and capacitors in the feedback loop. A digital hardware cannot make integration in the same way as in analog. Therefore a transformation of the integrator has to be done.

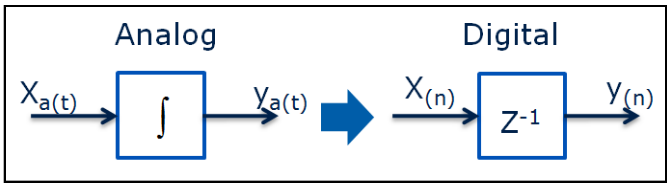


Figure 8.7 : Analog and digital integrator

Equation 8.14 describes how the output of the analog integrator is calculated.

$$y_{a(t)} = \int x_{a(t)} dt$$

Equation 8.14

To get a digital integrator the variable $y_{a(nT)}$ has to evaluated. This variable can be calculated as described in Equation 8.15.

$$y_{a(nT)} = y_{a[(n-1)T]} + \int_{(n-1)T}^{nT} x_{a(t)} dt$$

Equation 8.15

To find the value *nT* a approximation of the integral in Equation 8.15 has to be done.

The initial value in Equation 8.15 is the same but the value of the integral has to be calculated by taking the average value and multiplying by the interval as it is shown in Equation 8.16. The capital *T* is a small interval which has to be small enough that the spectrum is the same as in the baseband of the sampled signal.

$$y_{a(nT)} = y_{a[(n-1)T]} + \frac{1}{2}\frac{T}{}\left[x_{a(nT)} + x_{a[(n-1)T]}\right]$$

Equation 8.16

The differential equation for the integrator in the digital domain can be written as represented by Equation 8.17.

$$y_{(n)} - y_{(n-1)} = \frac{T}{2}\left[x_{(n)} + x_{(n-1)}\right]$$

Equation 8.17

By taking the z transformation to Equation 8.17 Equation 8.18 is given.

$$\frac{Y_{(z)}}{X_{(z)}} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$$

Equation 8.18

Equation 8.18 shows the transfer function of a digital integrator. This function has to correspond to *1/s* which describes the integrator in the analog domain. In other words, to transform a filter function into the digital domain, *s* has to be substituted by the reciprocal value of the right side from Equation 8.18 as shown in Equation 8.19.

$$H_{(z)} = H_{a(s)}\Big|_{s=\frac{T}{2}\frac{1-z^{-1}}{1+z^{-1}}}$$

Equation 8.19

This transformation from the s to the z domain is called bilinear transformation because the transformation is done with a rational fraction function with two linear polynomials. Considering that the analog transfer function is a useful transfer function and the transformation of the *Ha(s)* gives a useful transfer function in the digital domain, the first condition is that *H(z)* must be stable.

Therefore *z* in terms of *s* has to be calculated as given by Equation 8.20

$$z = \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}}$$

Equation 8.20

The magnitude of *z* is calculated in Equation 8.21.

$$|z| = \left|\frac{1+\frac{\sigma T}{2}+j\frac{\Omega T}{2}}{1-\frac{\sigma T}{2}+j\frac{\Omega T}{2}}\right|$$

Equation 8.21

Equation 8.22 shows how the magnitude is calculated.

$$|z| = \sqrt{\frac{\left(1+\frac{\sigma T}{2}\right)^2+\left(\frac{\Omega T}{2}\right)^2}{\left(1+\frac{\sigma T}{2}\right)^2+\left(\frac{\Omega T}{2}\right)^2}}$$

Equation 8.22

By evaluating the σ in Equation 8.22 the following correlation can be done.

If σ is negative, the nominator in Equation 8.22 is less than the denominator. Therefore the absolute value of z is less than 1. That means the imaginary axes from the left side in Figure 8.8 are transformed to the unit cycle shown in the right side of Figure 8.8. In case that σ is equal to zero the amount of z will be 1 which corresponds to a point on the unit cycle. And if the σ value is greater than zero, the amount of z will be greater than one. If *Ha(s)* is useful it must have all its poles in the left half plane. So the poles will be transformed inside the unit circle. The imaginary axis σ =0 is transformed to the unit cycle and the right half plane is transformed outside the unit circle. Therefore it is a stable transformation.

These correlations are also shown in Figure 8.8.

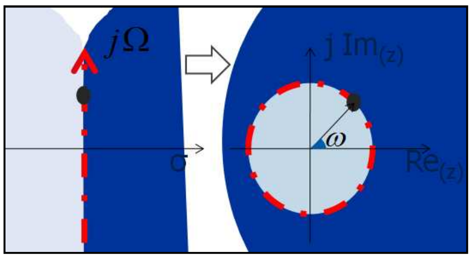


Figure 8.8 : Transformation of the s-plane into the z-plane

In particular, σ = 0 leads to $|z| = 1 \Rightarrow z = e^{j\omega}$. That means that a point on the imaginary axis at the left side of Figure 8.8 is transformed to a point at the right side in such way, that the angle is ω. This leads to Equation 8.23.

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \Rightarrow j\Omega = \frac{2}{T}\frac{1-e^{-j\omega}}{1+e^{-j\omega}}$$

Equation 8.23

With a little bit of manipulation Equation 8.23 can be calculated in a term of real quantity as shown in Equation 8.24.

$$j\Omega = \frac{2}{T}\frac{e^{-j\frac{\omega}{2}}}{e^{-j\frac{\omega}{2}}}\frac{2j\sin\left(\frac{\omega}{2}\right)}{2\cos\left(\frac{\omega}{2}\right)} = j\frac{2}{T}\tan\left(\frac{\omega}{2}\right)$$

Equation 8.24

Therefore the capital omega can be calculated as follows.

$$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$$

Equation 8.25

From Equation 8.25 Figure 8.9 can be derived. This figure shows the following relations.

$$\Omega > 0 \rightarrow 0 < \omega < \pi$$

Equation 8.26

and

$$\Omega < 0 \rightarrow \pi < \omega < 2\pi$$

Equation 8.27

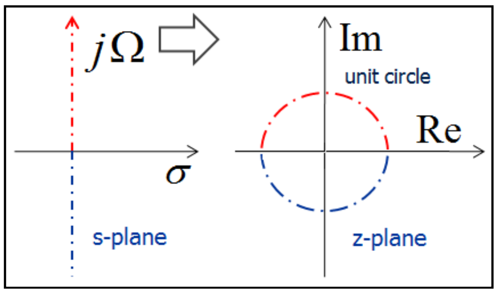


Figure 8.9 : Transformation of the imaginary axis

Equation 8.26 describes that the upper axis from the left side of Figure 8.9 goes to the upper side of the unit circle at the right side in Figure 8.9. And the upper side corresponds to the upper side of the unit circle. This relationship is a non linear relationship.

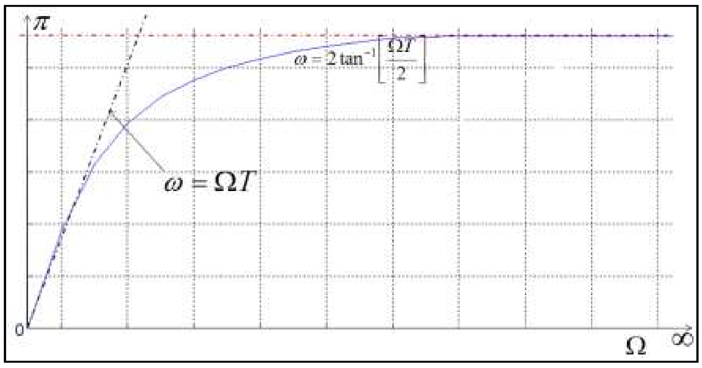


Figure 8.10 : Warping

Figure 8.10 shows the relation between small omega and capital omega. This figure shows that the non linear relation from the BLT and the linear part for small frequencies of the IIT. There is no aliasing in BLT because the total transfer function is being transformed. Figure 8.10 demonstrates also the so called warping effect. That means that an infinite scale Ω from 0 to infinite is compressed to a finite scale from 0 to pi. This advantage of frequency warping has to be compensated by pre-warping.

By designing a filter the specification has to be defined before.

Therefore the coefficients $\delta_p, \delta_s, \omega_p, \omega_s$ have to be defined for a low pass filter.

A typical specification for a low pass filter is shown in Figure 8.11.

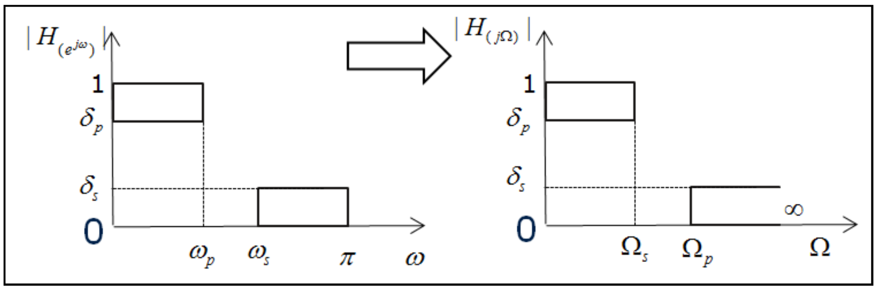
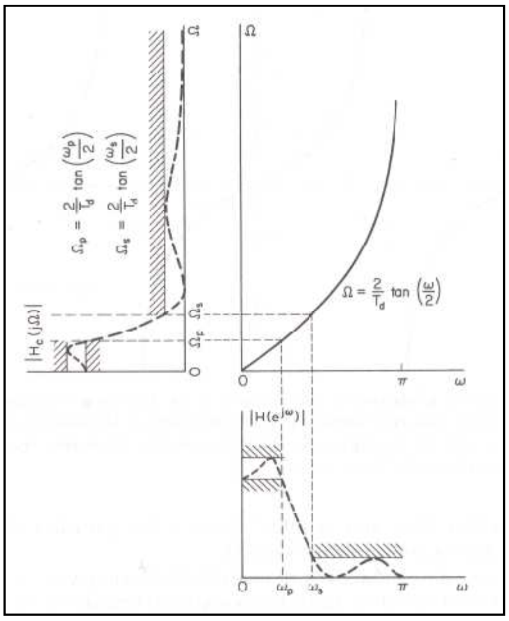


Figure 8.11 : LP filter specification

The variables ∂_p and ∂_s are invariant but the small omegas has to be transformed to the capital omegas by Equation 8.25 to get the specification of the analog filter. By using formula *s* = 2 / *t1* - *z*^-1 the warping in the other direction is done and gives the correct specifications. This relation which has been done analytically is contained in the diagram of Figure 8.12.



$|H_c(j\Omega)|$
$\Omega_p = \frac{2}{T_d}\tan\left(\frac{\omega_p}{2}\right)$
$\Omega_s = \frac{2}{T_d}\tan\left(\frac{\omega_s}{2}\right)$
Ω
$\Omega = \frac{2}{T_d}\tan\left(\frac{\omega}{2}\right)$
π
ω
$|H(e^{j\omega})|$
ω_p
ω_s

Figure 8.12 : Frequency Transformation [2]

In order to calculate the required order of the filter the Equation 8.29 is used.

$$\frac{\Omega_s}{\Omega_p} = \frac{\tan\left(\frac{w_s}{2}\right)}{\tan\left(\frac{w_p}{2}\right)}$$

Equation 8.28

Therefore the required order can be calculated as shown in Equation 8.29 and the cut off frequency is calculated by using Equation 8.30 for a Butterworth filter design.

$$N_B = \frac{\log 10\left(\sqrt{\frac{\frac{1}{\partial_s^{\ 2}} - 1}{\frac{1}{\partial_p^{\ 2}} - 1}}\right)}{\log 10\left(\frac{\Omega_S}{\Omega_p}\right)}$$

Equation 8.29

$$\Omega_C = \frac{\Omega_p}{\left(\frac{1}{\partial_p}^2 - 1\right)^{\frac{1}{2N_B}}}$$

Equation 8.30

[2]

9 Hardware

The block diagram in Figure 9.1 shows the components of the hardware setup. Dependent on the application there are two different circuits these will be described in detail below.

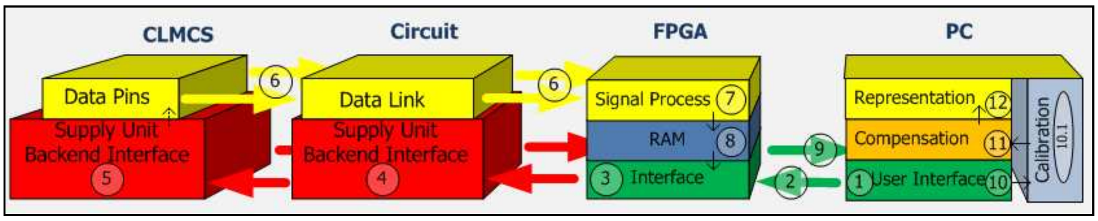


Figure 9.1 : Components of the used hardware.

The most important block is the CLMCS test chip. The current sensor delivers the data which includes information about the measured current. The current as the functionality of the interfering field rejection will be represented at the end of this working flow on a PC monitor. The chip has to be supplied by an individually developed modulation circuit block. Secondary to data connection and power supply, the circuit has the ability to do the amplitude modulation of the CLMCS supply. This modulation allows communicating with the digital interface of the chip and allows setup the sensor in different operating modes. The interface allows also reading out the register from the sensor. The only way to setup the CLMCS in to its ideal operating state is to set the registers in the chip by modulating the supply from the test chip. The supply modulation, respectively the protocol will be described below. As described in detail in the software chapter, the FPGA on the micro module buffers the data from the CLMCS and provides the communication between the current sensor and the computer.

The tasks at the PC are split into four parts. The first task of the PC is the communication with the FPGA. Therefore a USB interface has to be available at the PC's hardware to make control settings on the FPGA as well as supplemental settings on the CLMCS by sending the corresponding commands to the FPGA. Furthermore the calibration respectively, the compensation of the measured data as well as the visual representation of the current value has to be realized with the PC.

Figure 9.1 describes the process of measuring the applied current with the sensor and the peripheral hardware. At first the correct settings for the CLMCS are sent once only to the FPGA by the PC. After receiving the setting commands from the PC the backend interface of the FPGA will control the supply modulation circuit. The circuit will modulate the supply of the test chip to set the current sensor into a stable state. Secondly the supply modulation circuit is connects the data lines of the sensor with the FPGA. After the FPGA receives the data from the current sensor, the signal processing of the raw data will be done inside the FPGA on the micro module. The filtered and processed data of the three channels from the current sensor will be buffered at the RAM on the MM. Afterwards the FPGA will send the stored data to the PC via the USB interface. If a calibration has been done before, the PC has stored the coefficients for the compensation of the received data. Finally the calculated current will be represented at the monitor by the PC.

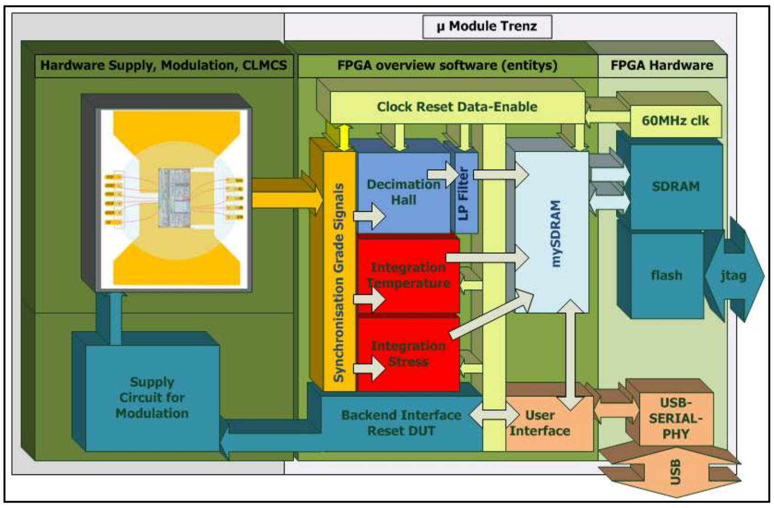


Figure 9.2 : Block diagram of measurement hardware and process blocks of the micro module

Figure 9.2 shows the main parts of the hardware which has been developed to measure the current with the linear Hall test chip. The left side represents the CLMCS connected with the developed supply modulation circuit. The right side shows the modules of the MM as the process blocks inside the FPGA.

9.1 CLMCS Core Less Magnetic Current Sensor

The current sensor is a derivative from the TLE4998 [17] product which is also a linear Hall sensor. The layout of the current sensor was built with three TLE4998 derivatives yet without any RAM. It is not possible to save parameters for calibration or startup setups on the test chip as it can be done at the TLE4998 magnetic sensor. In contrast to the TLE4998, which only consists of one data channel, the current sensor has three Hall sensors which are all working simultaneous on the chip. The three parallel working sensors are the precondition which enables the rejection of the interfering field. To utilize the magnetic sensor to measure the magnetic field, which is generated by a current flow, one sensor would be sufficient. However in the majority of cases in a field of application there are a lot of cross talking magnetic fields so a sensor capable of influence field rejection will be advantageous. The coreless magnetic current sensor does not need a mechanical protection to compensate against interfering fields. As long as the interference field is homogenous or has a linear gradient, the signals from the three Hall plates can be calculated by using a differential form which ensures the elimination of the superposed. Equation 9.1 describes how the compensated signal has to be calculated at the end of the signal processing.

$$Out = Hall_{middle} - \left(\frac{Hall_{left} + Hall_{right}}{2} \right)$$

Equation 9.1

Because of non linear effects in the semiconductor, caused by stress and temperature, the digital values of the Hall plates have to be corrected from this impact. Equation 9.1 describes how the interfering fields will be eliminated. The compensation process is described in the compensation chapter. Figure 9.3 represents how the 'Triple Hall Principle' is used on the CLMCS chip. At the bottom of Figure 9.3 the dashed line marks the outline of the current sensor on the PCB above the current rail. The current rail consists of a copper conductor with a thickness of 400µm. Because of the focused magnetic field above the slits in the current rail, which is produced by the current, it is important to place the Hall plates close and exactly to the slits in the copper conductor to ensure that the Hall sensors are detecting the field cone. Figure 9.3 shows the direction of the magnetic field at the specific slits in the current rail dependent on the applied current. The technical current flow will be represented by the red line in Figure 9.3. In a case where the current flow runs from the right side to the left side, the direction of the magnetic field would travel from the top to the bottom at the left and right slit. In contrast to the left and right slit, the direction of the magnetic field on the middle channel shows the opposite direction. The value of the magnetic field at the slits, which is produced by the current flow on the current rail, is also represented at the top in Figure 9.3. Dependent on the accuracy of the placements from the Hall plates on the current rail and the pitch between the chip and the copper conductor, the chip will detect a magnetic field in linear proportion to the current at a specific temperature. The blue marks in Figure 9.3 are indicating the magnetic field caused by the current. The gray mark in Figure 9.3 represents an interfering field. The top of Figure 9.3 shows the direction of the interfering field and the lower drawing in Figure 9.3 shows the value of the magnetic field in mT at each Hall plate. As long as the order of the gradient of the interfering field is smaller or equal to one, the noise field will be eliminated by using the Equation 9.1. Figure 9.3 also shows the values of the interfering fields. By subtracting the mean value of the outside channel from the middle channels, the value of the interfering field will be zero. That means, only the magnetic field caused by the current, will be measured and allocated. Figure 9.3 shows realistic ratios between the magnetic field and applied current for the PCB layout. The drawing also gives an indication of the mechanical proportions. Figure 9.3 serves only to show the concept of the chip and is not to scale. Due to the layout of the current rail, the current will be squeezed to run close to the slits. The lower drawing in Figure 9.3 illustrates that the current density at the middle slit is higher than the density at the left and right channel. Due to the higher current density at the middle slit, dependent on the mechanical design of the current rail, the value of the magnetic field, which is proportional to the current, will have its maximum on the middle channel. Therefore the middle Hall plate detects a larger value of magnetic field as compared to the other channels. Thus the Hall sensitivity of the middle channel is higher than the sensitivity of the others. The difference of the sensitivity between the middle Hall plate and the outer ones is approximately 20 percent at this layout.

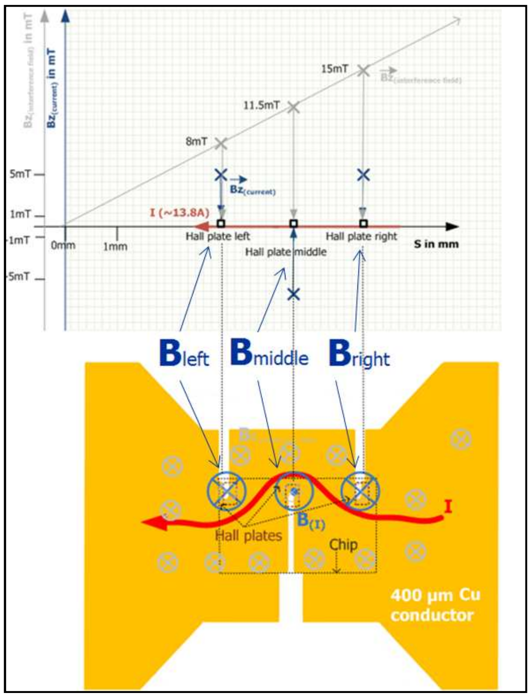


Figure 9.3 : Triple Hall Principle; magnetic field produced by load current superposed by an interfering field at the slits in the conductor

Figure 9.4 shows the CLMCS with all of the data pins. In addition Figure 9.4 shows the chip with all of its functional parts. As mentioned before it is important to get the data from the chip using a parallel method to ensure fast data analysis. To guarantee a parallel time synchronous data process the chip consists of three linear magnetic sensors working in parallel. Figure 9.5 gives an overview of the current sensor modules.

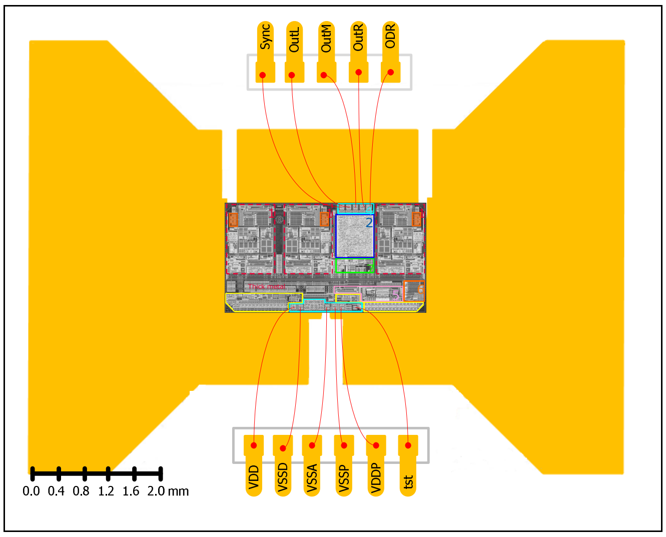


Figure 9.4 : CLMCS on PCB with bonding plan

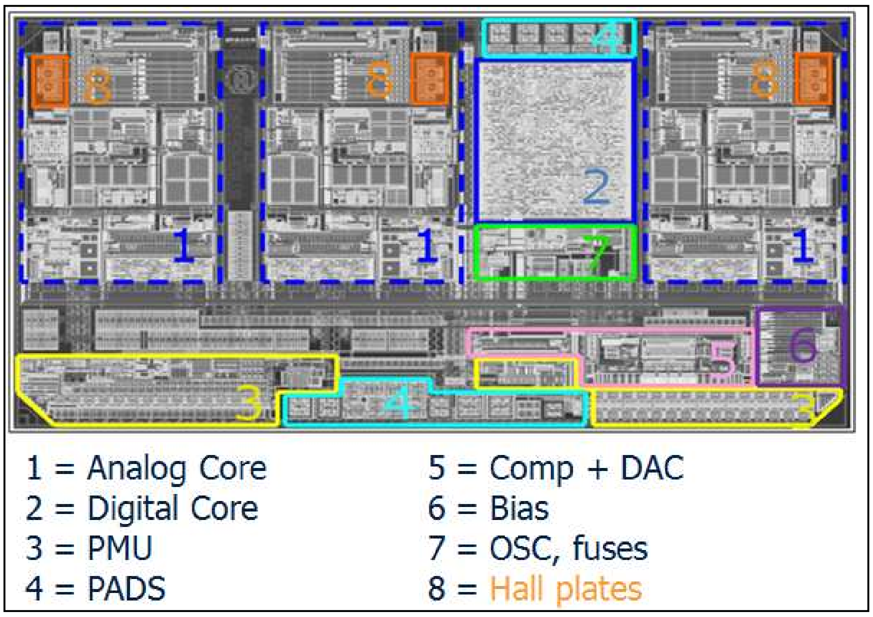


Figure 9.5 : Design overview of the CLMCS test chip [11]

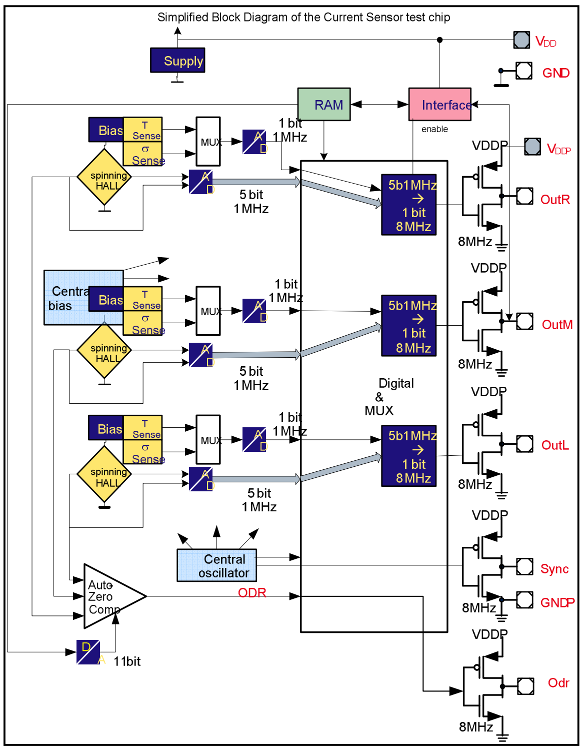


Figure 9.6 : Simplified block diagram of the current sensor [11]

Figure 9.6 illustrates the block diagram of the current sensor. It describes the main parts of each channel. As shown in the block diagram, each of the three channels consists of spinning Hall plates which will be supplied by a bias system. A spinning principle guarantees a small offset error.

Moreover every channel has a temperature and a mechanical stress sensor which responds to the level of the bias voltage. The bias voltage can be adjusted via the backend interface of the chip. Therefore seven bits are available to setup the value via the interface. There are also four further bits for the fine tuning of the bias level for the temperature and stress sensor. The test pin which is shown in Figure 9.4 allows measuring the adjusted bias voltage. Therefore the multiplexer which connects the output of the bias to the test pin has to be adjusted with the corresponding command via the backend interface. The bias setting is a consequential setup for the chip characteristic. The chapter bias setting and operating mode describes how the ideal bias setting will be found. The analog output value of the Hall sensor will be converted by a continuous time sigma delta ADC. Additionally the analog output from the temperature and stress sensors is converted in a digital signal by an analog to digital converter. Because of the inertia tendency of the physical quantity of the temperature, the output of the temperature and the stress sensor can be multiplexed. Only one ADC will be used for each temperature and stress unit for each channel on the current sensor. The T/S-ADC delivers a one bit signal with a one MHz update rate. The sigma delta ADC from the Hall plates sends a five bit signal also with a frequency of one MHz. The five bit Hall signal and the one bit multiplexed temperature and mechanical stress signal of all three channels will be sent to the digital part of the chip. The digital unit prepares the signals from the Hall, temperature and stress sensor for each channel in a seven bit signal and sends them to the output. The digital unit also generates a signal for the synchronization at the output. The so called sync signal contains the time information for the correct sampling time of the three output signals left, middle and right.

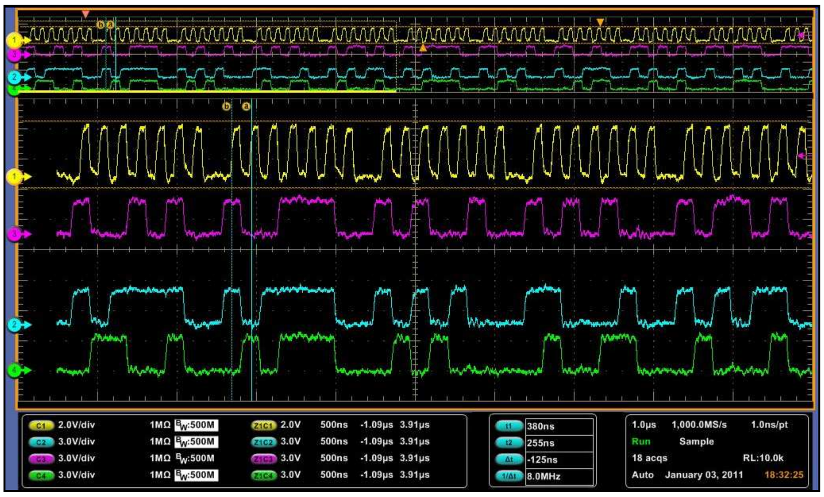

```
C1  2.0V/div   1MΩ  BW:500M   Z1C1 2.0V  500ns  -1.09µs  3.91µs     t1   380ns     1.0µs   1,000.0MS/s    1.0ns/pt
C2  3.0V/div   1MΩ  BW:500M   Z1C2 3.0V  500ns  -1.09µs  3.91µs     t2   255ns     Run        Sample
C3  3.0V/div   1MΩ  BW:500M   Z1C3 3.0V  500ns  -1.09µs  3.91µs     Δt   -125ns    18 acqs              RL:10.0k
C4  3.0V/div   1MΩ  BW:500M   Z1C4 3.0V  500ns  -1.09µs  3.91µs     1/Δt 8.0MHz    Auto   January 03, 2011   18:32:25
```

Figure 9.7 : Measured digital output signals from the CLMCS

Figure 9.7 shows the three data signals and the synchronization signal with a one MHz gap. Channel one represents the synchronization signal. The data will be captured by the FPGA at every rising edge of sync signal.

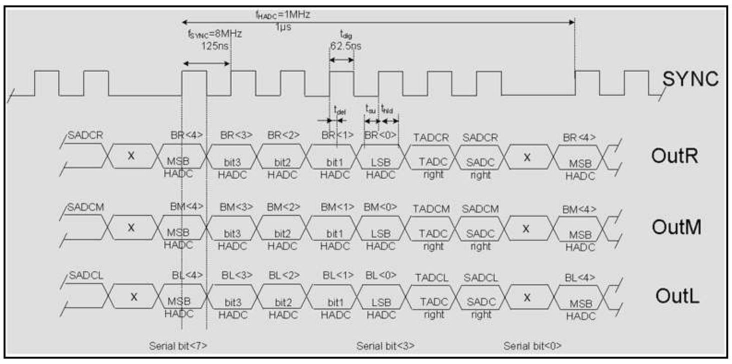


Figure 9.8 : Output protocol of the CLMCS [11]

Figure 9.8 shows the timing protocol of the output data from the current sensor. The protocol and communication will be described in detail in the backend interface chapter.

The sync signal is used for the synchronization of the output signals at the channels 'OUTR', 'OUTL' and 'OUTM' with the FPGA. Each of the three outputs sends a nonstop eight MHz signal. One data set consists of a five bit Hall data and a one bit temperature as well as a one bit stress signal as previously mentioned. For a fast current overdrive detection there is an extra pin at the CLMCS. The so called 'ODR' pin is a digital output of the sensor which indicates if a current overdrive occurs. Therefore the reference of the fast comparator can be set via the interface. That means, that the overdrive current level can be programmed by the interface as shown in Figure 9.8 one data set consists of seven bits. The gap in the 'sync' signal which can be seen every one µs, signals the beginning of a new dataset at the output signal of the current sensor. After every synchronization gap, a new dataset starts. Every data record begins with the most significant bit of the 5bit Hall signal and ends with the stress bit. The second last bit at position two includes the temperature information. For further signal processing like buffering, filtering or decimation, a special hardware had to be developed in the course of this thesis. The main requirement was to ensure a fast and parallel working data process. Therefore a FPGA was chosen for all continuous signal processes.

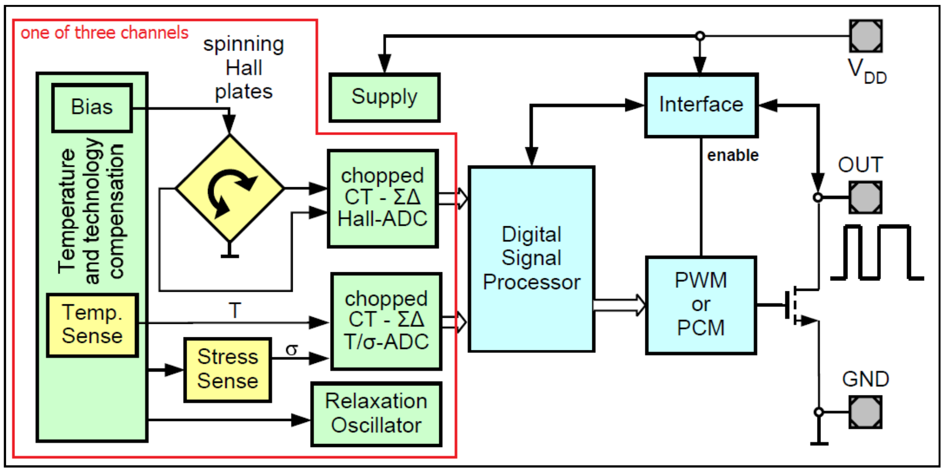


Figure 9.9 : Block diagram of the current sensor (one channel) [9]

Figure 9.9 indicates the block diagram of the sensor. Only one of three channels is. The principle of the stress sensing is described in the bias setting chapter. To convert the temperature and stress signal into the digital domain a one bit sigma delta converter is shared. The output of the spinning Hall plates is connected to a 3rd order continuous time sigma delta ADC. The continuous time sigma delta has the advantage that no aliasing occurs.

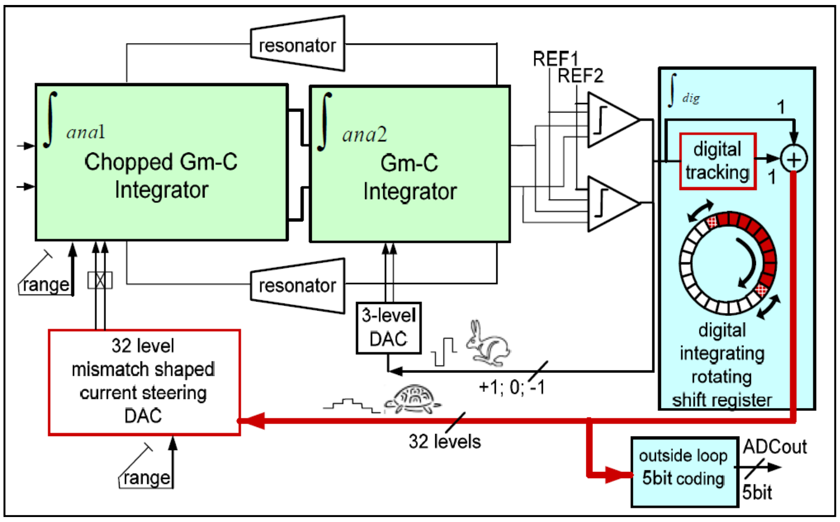


Figure 9.10 : Continuous Time Sigma Delta Block Diagram [11]

Figure 9.10 shows the block diagram of the implemented sigma delta converter. Figure 9.10 indicates the digital tracking loop as well as the range switches to increase the robustness. The first two integrator stages are analog integrators. The third stage of the ADC consists of a digital integrator. In this area also the digital integrating rotation shift register can be seen in Figure 9.10. This rotating register allows increasing the accuracy because the reference currents for the DAC are shifted. Therefore the performance will be increased by averaging the mismatch of the current sources. The inner feedback loop has only a three level feedback loop to ensure a fast feedback loop. The frequency of the shift register is represented in the frequency domain in Figure 9.12. This peak has no impact to the wanted signal because the peak from the shift register is not in the band of interest. In Figure 9.12 also the noise shaping effect of the sigma delta ADC can be seen. The peak at 22kHz in the spectrum of Figure 9.12 depends on the over sampling ratio, respectively the gain band width, which is dependent on the feedback factor of the ADC.

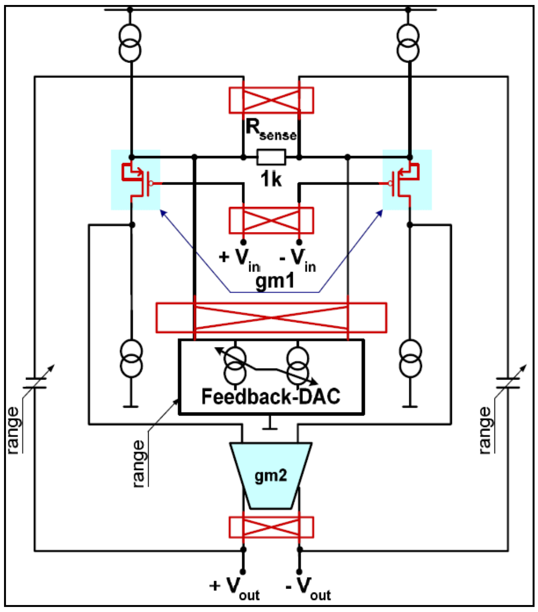


Figure 9.11 : Serial connection of 2 analog integrators with chopper stages [11]

Figure 9.11 shows the serial connection of two analog integrators of the sigma delta ADC. To increase the accuracy, the signal is chopped in the integrator. Therefore the mismatch of the threshold voltage (*U_{GS}*) of the two transistors dependent on the temperature increases by averaging. To avoid the influence of noise in the low frequency domain, dependent on the serial connection of the direct current amplifier, the input voltage is converted in an alternating signal. Therefore a chopper at the input stage will chop the input DC signal into an AC signal. This signal is amplified by the tranconductance and again chopped after the amplifier stage. Thus the amplified output signal will be a DC signal again. To ensure a correct working flow, the modulation frequency of the input stage has to be exactly the same as the frequency of the demodulation stage after the amplifier. The disadvantage of these amplifiers is that the influence of noise with a frequency close to the chopping frequency is given. This impact does not have an influence if the chopping frequency is sufficient higher than the highest frequency of the wanted signal.

The input resistor of the integrator stages is replaced by an operational transconductance amplifier which converts the input voltage from the Hall plate into a current. The capacitors in the current feedback are used to prevent leakage currents and parasitic capacitors from effecting the transfer function [9]. The current of the feedback-DAC depends on the band gap. Due to this current, a voltage drop at the resistor R_{sens} can be detected obtaining a precisely compensation of the sensitivity from the Hall plate.

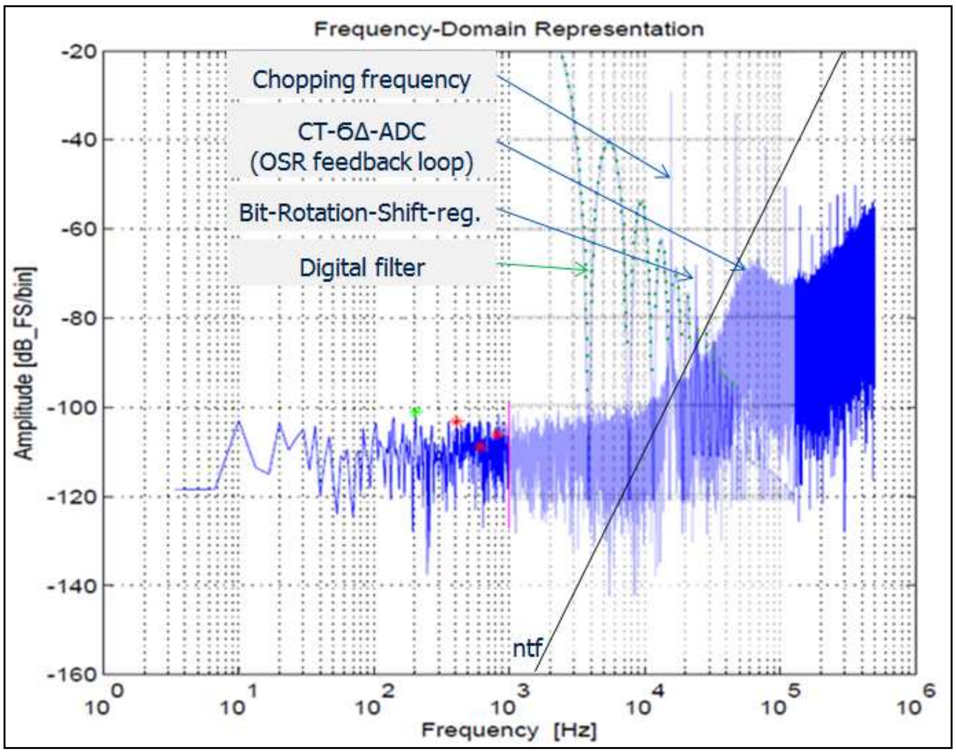


Figure 9.12 : Spectrum analyzes of the Hall data

Sigma delta principle and noise shaping effect

Due to different contaminations, respectively inhomogeneous mechanical dimensions in the Hall plates or punctual stress, originated by filler parts in mold compound of plastic package, each Hall element shows a different offset. To pre-compensate the offset dependent on the assembling, the spinning chopping principal can be used. Therefore the Hall plate is chopped. The chopping principal is the following. In the first step the Hall plate is supplied from the Hall bias system by connecting the clamps 1 and 2 with the bias system. At the clamps 3 and 4 the voltage, which is superposed by the offset signal, dependent on the magnetic field can be measured.

$$U_1 = U_{(B)1} + U_{(Hall\_offset)1}$$

Equation 9.2

In the next phase the Hall plate is switched to 90°. Thereby the Hall plate is supplied via the clamps 3 and 4 and the produced voltage can be measured at 1 and 2. Therefore the measured Hall voltage U2 is different in the sign to the measurement from phase 1. But the offset voltage from the Hall plate shows the same sign as in the measurement before.

$$U_2 = U_{(B)2} - U_{(Hall\_offset)2}$$

Equation 9.3

This effect can be utilized to reduce the offset signal from the Hall plate. Thereby the difference of the two measured voltages has to be calculated and divided by two shown in Equation 9.4.

$$U_{chopped} = \frac{\left[\left(U_{(B)1} + U_{(Hall\_offset)1}\right) + \left(U_{(B)2} - U_{(Hall\_offset)2}\right)\right]}{2}$$

Equation 9.4

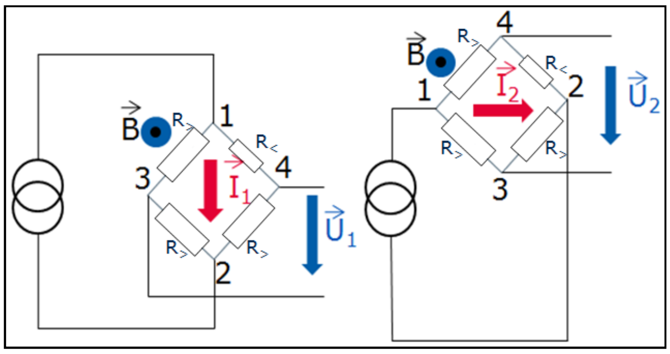


Figure 9.13 : Chopping principle of the Hall plates

9.2 Micro Module

The XILINX XC3S1000 FPGA [8] has been used for development of the demonstrator tool and the measurement hardware for the laboratory. This FPGA is installed on the micro module TE0146-00 from the company 'trenz-electronic'. There is also a synchronous DRAM [7] on this micro module. In addition there is a USB-Phy [18] installed on the module which, converts a serial RS232 protocol into a USB protocol. The USB-Phy supports USB1/2 full– and high speed mode. This micro module has been used to develop the hardware. Also a carrier board for the TE0146-00 module from the 'trenz electronic' company has been used to program the FPGA. Furthermore the carrier board has been used to connect the measurement hardware with the micro module in the laboratory setup. As described, some modulation of the supply from the current sensor has been done to bring the test chip into a stable state. Therefore additional hardware has to be developed. The so called supply modulation circuit has to be designed in the same mechanical size as the micro module to ensure a small and compact demonstrator tool. The identical sizes of the modules allow stacking of the modules.

Figure 9.14 shows all pins which are available on the used FPGA. Some of them are already in use connecting the FPGA with the peripheral hardware such as the USB–Phy or the SDRAM on the micro module.

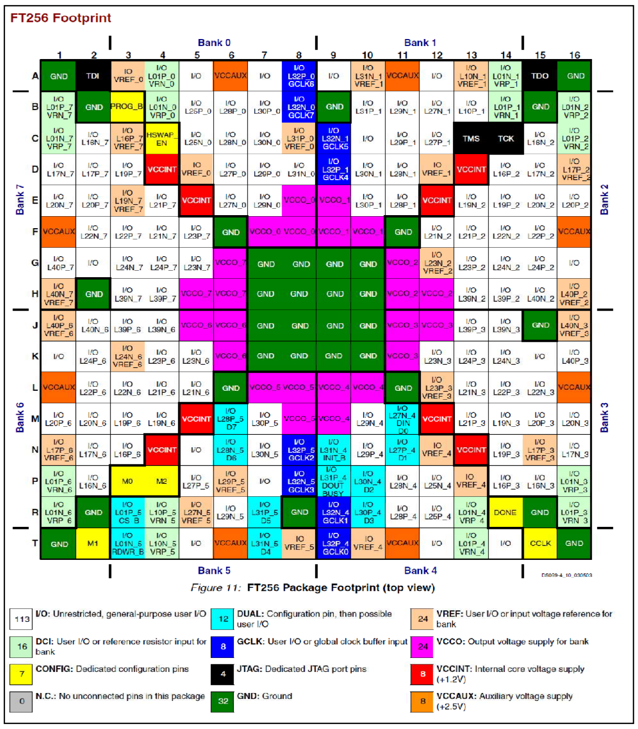

FT256 Footprint
Bank 0
Bank 1
Bank 2
Bank 3
Bank 4
Bank 5
Bank 6
Bank 7
Figure 11: FT256 Package Footprint (top view)
113 I/O: Unrestricted, general-purpose user I/O
12 DUAL: Configuration pin, then possible user I/O
24 VREF: User I/O or input voltage reference for bank
16 DCI: User I/O or reference resistor input for bank
8 GCLK: User I/O or global clock buffer input
24 VCCO: Output voltage supply for bank
7 CONFIG: Dedicated configuration pins
4 JTAG: Dedicated JTAG port pins
8 VCCINT: Internal core voltage supply (+1.2V)
0 N.C.: No unconnected pins in this package
32 GND: Ground
8 VCCAUX: Auxiliary voltage supply (+2.5V)

Figure 9.14 : Footprint of the Xilinx FPGA (FT256) [8]

In order to communicate with the current test chip, eight further free I/O ports have to be declared. Table 9.1 shows which signals have to be connected with the hardware to ensure the correct process flow.

Communication pins between FPGA and Modulation Circuit(MC) / CLMCS		
Naming	connection	Usage
Left_i	CLMCS – MC - FPGA	Input left channel from test chip
Middle_i	CLMCS – MC - FPGA	Input middle channel from test chip
Right_i	CLMCS – MC - FPGA	Input right channel from test chip
Synch_i	CLMCS – MC - FPGA	8 MHz synchronization signal
Odr_i	CLMCS – MC - FPGA	Input from test chip for over current
MODN	MC - FPGA	Output to modulate VDD for Backend
RES_DUT	MC - FPGA	Output to reset the Chip (analog)
RES_VDD	MC - FPGA	Output to reset the Chip (digital)

Table 9.1 : Pin content between Micro Module and CLMCS / Modulation Circuit

There is only one Board to Board connector (B2B) on the micro module because the right side of the module is used for the SDRAM. For this reason the I/O ports are limited to the free pins on the left connection port.

Figure 9.15 illustrates the left board connector of the micro module. All pins are connected with the FPGA on the MM.

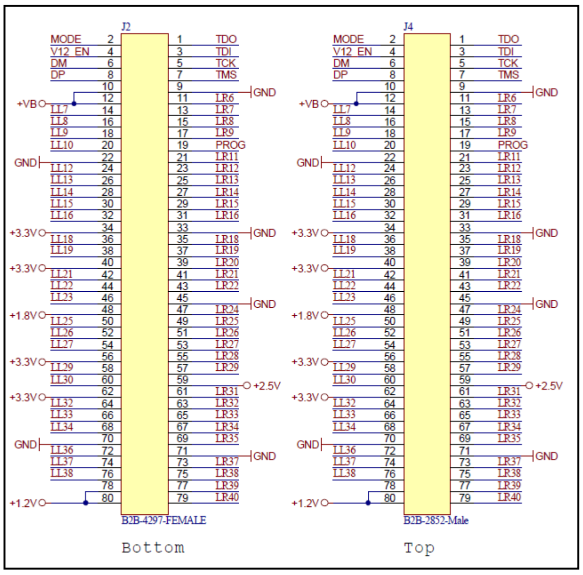


Figure 9.15 : Board to Board Connector J2 of the Micro Module [19]

9.2.1 Programming the FPGA

The synthesis has been done with the Xilinx ISE 10.1 software. In order to load the program code into the FPGA, the carrier board and the JTAG Programmer need to be used. Figure 9.16 shows a drawing of the carrier board which is also used to connect the FPGA to the measurement hardware in order to perform calibration measurements in the laboratory. The connectors *J1* and *J3* will be used to program the FPGA on the micro module via the JTAG protocol.

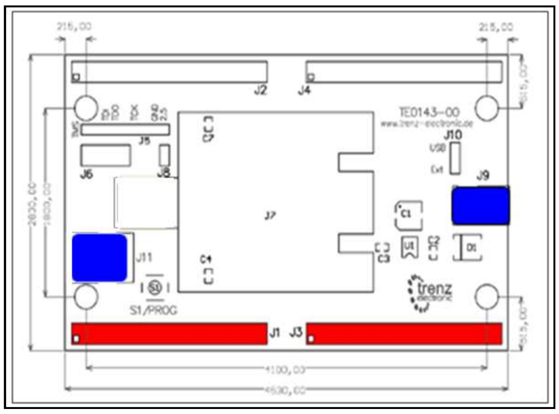


Figure 9.16 : Carrier – and Programming Board of the Micro Module

Figure 9.17 demonstrates how the USB - JTAG Programmer has to be connected to the carrier board. In order to program the FPGA via the USB -JTAG cable it is necessary to supply the micro module either with a five volt mains adapter (J9) or via the USB connector on the micro module (J11). These connections are marked blue in drawing Figure 9.16. Table 9.2 contains the pin assignment to program the FPGA on the micro module which has to be connected to the carrier board via the B2B connector J2.

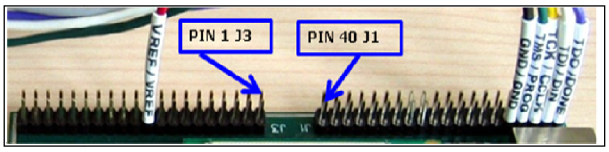


Figure 9.17 : Pin content to connect the JTAG-Programmer with the Carrier Board

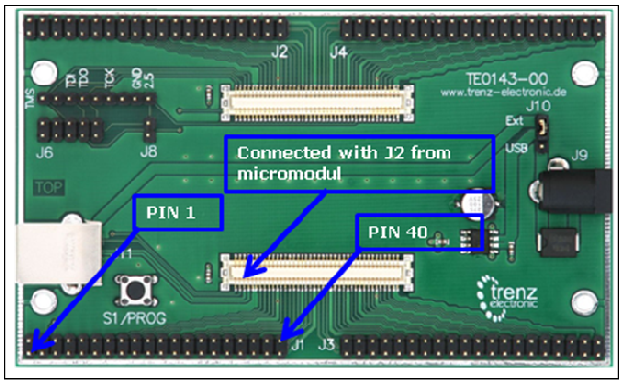


Figure 9.18 : Carrier Board of the Micro Module

Figure 9.18 shows the carrier board where the micro module has to be connected to program the FPGA on the MM. The connector *J2* in Figure 9.18 will be used to connect the micro module via the B2B connector.

Connector	**Pin**	**Signal**
J1	2	TDO
J1	4	TDI
J1	6	TCK
J1	8	TMS
J1	10	GND
J3	20	VREF
J11		Supply USB
J9		Supply ext.5V adapter

Table 9.2 : Pin assignment of the Carrier Board for programming the FPGA

9.3 Supply Modulation Circuit

A compact demonstrator tool has to be designed allowing demonstration of the correct function of the test chip. Therefore the supply–and modulation levels which are shown in Figure 9.19 have to be considered.

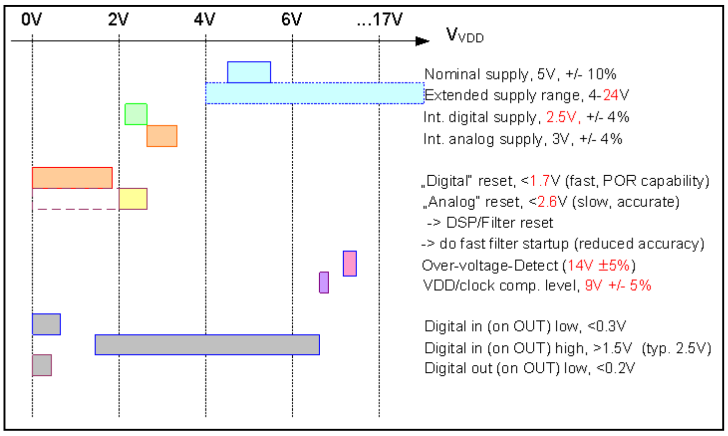


Figure 9.19 : Voltage levels used inside the current sensor [11]

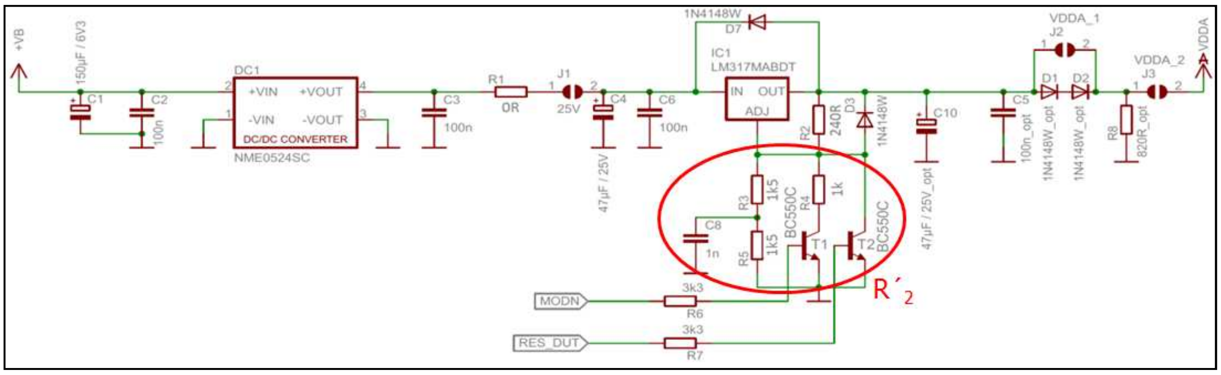


Figure 9.20 : Modulation Circuit for the analog chip supply

The demonstrator tool will be powered by the USB–interface of a PC. To ensure the power supply for the CLMCS from the demonstrator tool respectively, a communication between the backend interface of the current sensor and the demonstrator, the five volt power supply from the USB port has to be boosted to a higher level by a DC/DC converter. The DC/DC converter at the input stage of the circuit converts the five volt input voltage to 24 volt. As described in the backend interface chapter the demonstrator is responsible for power supply to the test chip and must also guarantee a defined reset to set the CLMCS in the test mode by modulating a defined startup sequence. To setup the chip in the test mode and for further communication, respectively setups, it is necessary to modulate the power supply from the chip.

Figure 9.21 shows a startup sequence modulated by the described hardware.

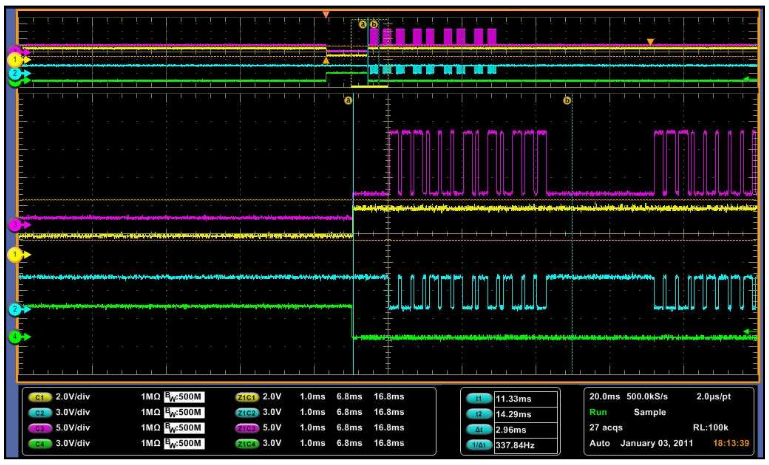


Figure 9.21 : Modulated Start sequence after reset on the CLMCS supply

The signals *MODN* and *RES_DUT* in Figure 9.20 are output signals from the FPGA and will be triggered by the backend interface of the FPGA. *MODN* is used to modulate the supply of the CLMCS. The *RES_DUT* signal will reset the chip. If the *RES_DUT* signal is set to high, respectively to 3.3 V, the transistor *T*2 in Figure 9.20 will become conductive and the output of the linear voltage regulator will be set to approximately 1.25V. The series connection of the diodes *D*1 and *D*2 at Figure 9.20 ensures a reduction of the voltage to nearly zero volts. *D*1 and *D*2 have an option of providing a lower reset voltage if required. Channel four in Figure 9.21 represents the *RES_DUT* signal from the interface. Channel two shows the status of the supply modulation signal *MODN*. Channel three represents the analog supply level of the chip. The digital supply is represented on channel one. It is possible to trigger a reset by sending the reset command to the FPGA via the USB interface. Figure 9.21 demonstrates the signals which are controlling the modulation hardware and the start-up sequence after the reset. Channel three represents the analog power supply level of the sensor. Channel two shows the status of the supply modulation signal *MODN*. Figure 9.21 clearly shows how the supply voltage will be modulated dependent on the *MODN* signal. After the reset, the *RES_DUT* signal at channel four returns to the low voltage status. If the supply modulation signals at channel two changes the status from 3.3 to 0 volts, the ratio of the potential divider at the output of the voltage regulator in Figure 9.20 will be changed.

Equation 9.5 shows how the voltage at the output of the voltage regulator will be calculated.

$$U_{out} = V_{ref} \cdot \left(1 + \frac{R_2'}{R_2}\right) + I_{ADJ} \cdot R_2$$

Equation 9.5

The value of the R_2' equivalent resistance in figure 7.15 depends on the U_{BE} voltage level of the transistors T1 and T2 which is controlled by the backend interface of the FPGA.

$$U_{out} = 1.25V \cdot \left(1 + \frac{3k\Omega}{240\Omega}\right) \approx 16.875V \Big|_{MODN=0V \& RES\_DUT=0V}$$

Equation 9.6

$$U_{out} = 1.25V \cdot \left(1 + \frac{750\Omega}{240\Omega}\right) \approx 5.15V_{|MODN=3.3V \& RES\_DUT=0V}$$

Equation 9.7

Figure 9.22 shows how the supply modulation of the VDD will be transformed into a SPI protocol inside the test chip.

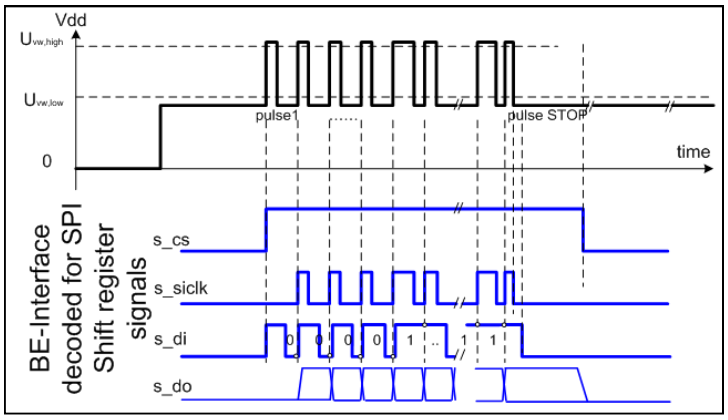


Figure 9.22 : Decoded SPI – Signal for the CLMCS Backend Interface [11]

Figure 9.23 demonstrates the circuit for the power supply of the digital part of the current sensor. The signal *RES_VDD* is also an output signal from the FPGA which will be triggered by the backend interface from the FPGA. The signal *RES_VDD* always has the same status as the signal *RES_DUT*. If this signal is set to high, the digital power supply *VDDP* of the current sensor will be set close to zero volts to reset the chip.

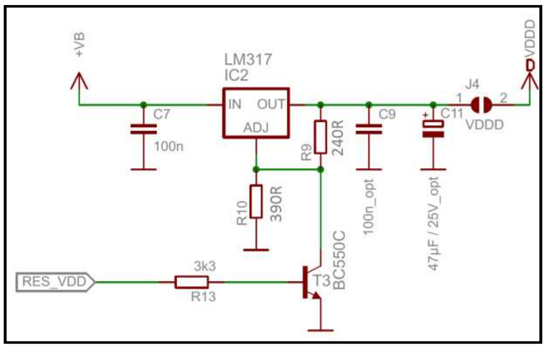


Figure 9.23 : Digital Supply Circuit of the CLMCS

The appendix includes all components of the described circuits as well as the list of materials.

9.4 Demonstrator Tool

Figure 9.24 describes how the circuit of the demonstrator tool is connected to the connector of the micro module. There is a different pin assignment at the B2B connector of the micro module dependent on the developed hardware.

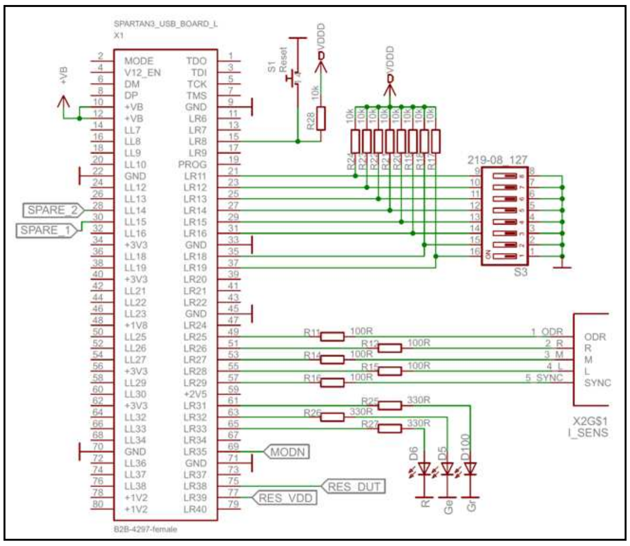


Figure 9.24 : Pin assignment between the Demonstrator Tool and the Micro Module

Table 9.3 gives a detailed description of the wiring of the board connector from the micro module if the FPGA is used for the demonstrator tool. The appendix includes the whole netlist of the FPGA for the demonstrator tool.

Connector (Micro Module)	Pin	Label	Signal	Signal name	Bank	Pin FPGA
J2	15	LR8	Reset	Rst_n_i	7	D3
J2	21	LR11	Diptaster	Diptaster_o<7>	7	E2
J2	23	LR12	Diptaster	Diptaster_o<6>	7	E1
J2	25	LR13	Diptaster	Diptaster_o<5>	7	F3
J2	27	LR14	Diptaster	Diptaster_o<4>	7	F2
J2	29	LR15	Diptaster	Diptatser_o<3>	7	G4
J2	31	LR16	Diptaster	Diptaster_o<2>	7	G3
J2	35	LR18	Diptaster	Diptaster_o<1>	7	G1
J2	37	LR19	Diptaster	Diptaster_o<0>	7	G2
J2	49	LR25	overdrive	Odr_i	6	J2
J2	51	LR26	Right channel in	Out_r_i	6	K2
J2	53	LR27	Middle channel	Out_m_i	6	K1
J2	55	LR28	Left channel in	Out_l_i	6	L3
J2	57	LR29	Synchronization	Sync_i	6	L2
J2	61	LR31	LED green	Test3_o	6	M3
J2	63	LR32	LED yellow	Test2_o	6	M2
J2	65	LR33	LED red	Test1_o	6	M1
J2	69	LR35	Modulate VDD	Vdd_clk_o	6	L5
J2	75	LR38	Reset analog	Supplyoffdut_o	6	N2
J2	77	LR39	Reset digital	Supplyoff_o	6	N3

Table 9.3 : Pin assignment for the demonstrator tool between FPGA and Micro Module

The demonstration tool for the current sensor consists of three main components. One part of the hardware is the micro module from Trenz-electronics. As shown in Figure 9.26, this module includes the Xilinx Spartan XCS1000 FPGA, the micron synchronous DRAM and the SMSC USB-Phy. The second part is the designed modulation circuit which allows the communication between the CLMCS and the micro module. The third part is a package which protects the hardware from mechanical impacts and makes handling of the hardware more comfortable for the user.

Figure 9.25 shows the modules which have been used for the demonstration tool.

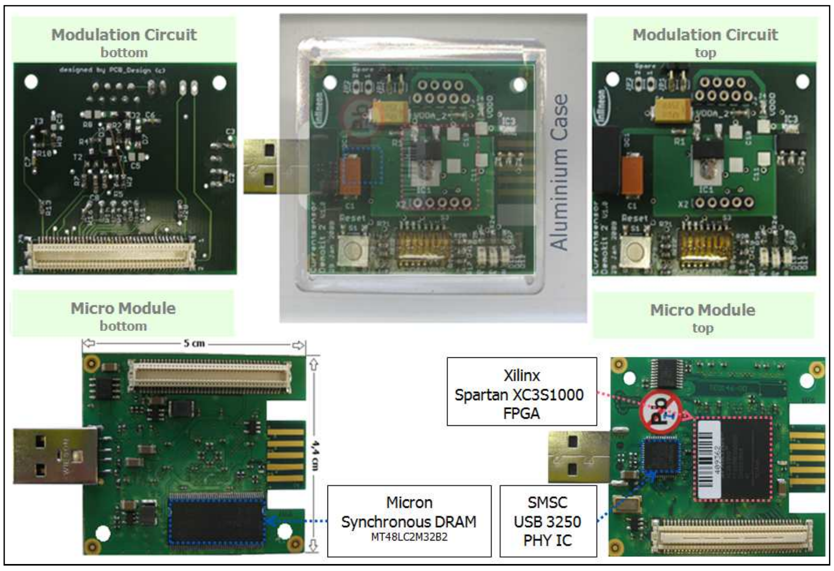


Figure 9.25 : Modules of the Demonstration tool

The modulation circuit is a two layer PCB with a size compatible with the micro module. In Figure 9.25 the dimensions of the micro module can be seen. The similar dimensions of the two boards allow them stacking together. The B2B connector on the boards avoids an incorrect mechanical connection between the boards because of notches on the connector. The stacked modules will be put into an aluminum case seen in Figure 9.25 and Figure 9.26.

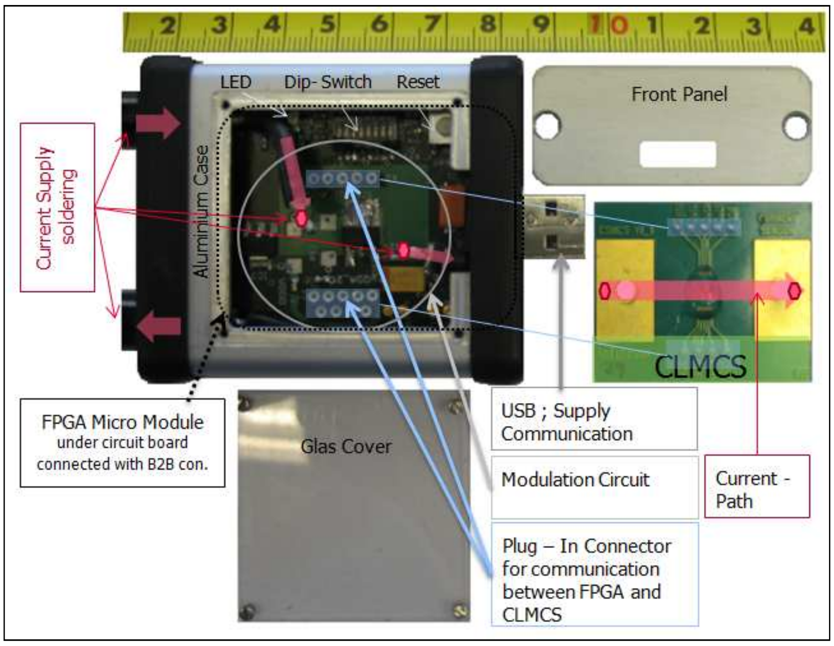


Figure 9.26 : Modules and Dimension of the Demonstrator

Figure 9.26 shows the hardware inside the aluminum case in the top view. On the upper side of the modulation circuit, the plug in connector for the current sensors can be seen in Figure

9.26. The connectors at the rear side of the package are wired to the connections of the current sensor where they have to been soldered to the current rail of the CLMCS. If the current sensor would be inserted into the demonstrator tool a glass cover will protect the sensor against mechanical impact. The front panel has a cut out for the USB connector of the micro module.

Figure 9.27 shows the connected modules with the current sensor on the top.

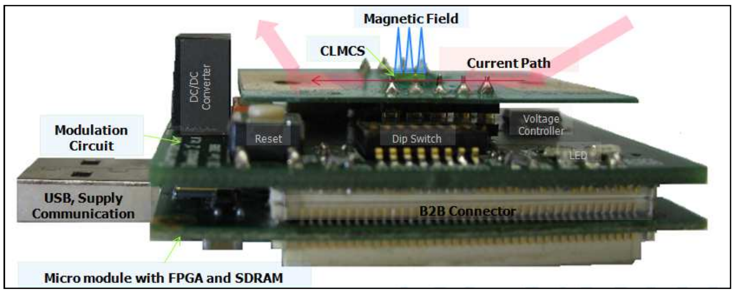


Figure 9.27 : Stacked Demonstrator modules with the connected CLMCS on top

9.5 Hardware for automated characterization

For the measurements in the laboratory the FPGA has to be connected with the so called measurement hardware. Therefore the pinning from the J2 connector of the micro module has to be changed for some signals. The carrier board connects the micro module with the measurement hardware via the J1 multi pin connector.

Table 9.4 gives an overview of all pins which have been changed in the net list of the FPGA if the measurement hardware for the IFX intern characterizations will be used.

Connector Micromodul	Pin	Label Name	Signal Name	Connector Board	Pin	Label Name	PIN FPGA
J2 / J4	37	LR19	overdrive	J1	38	LR19	G2
J2 / J4	35	LR18	Rigth channel in	J1	36	LR18	G1
J2 / J4	36	LL18	Middle chnnel in	J1	35	LL18	J4
J2 / J4	33	GND	GND	J1	34	GND	-
J2 / J4	34	3.3V	3.3V	J1	33	3.3V	-
J2 / J4	31	LR16	Left channel in	J1	32	LR16	G3
J2 / J4	29	LR15	Sync	J1	30	LR15	G4
J2 / J4	28	LL14	Reset	J1	27	LL14	F4
J2 / J4	26	LL13	Vdd_clk_o	J1	25	LL13	E4
J2 / J4	16	LL8	Reset dut	J1	15	LL8	D7

Table 9.4 : : Pin assignment for the measure hardware between FPGA and Micro Module

One measurement required in the laboratory is the application of a higher current on the test chip. Therefore the hardware and especially the supply connection for the load current has to be adapted. Thus the current sensor can also be used for verifications and characterizations in the laboratory. Hence a measure hardware with an adequate copper conductor with a small transition resistance will be necessary, to supply the circuit. Another requirement in the laboratory is the measurement over a wide range of temperatures. To make such measure routines it should be possible to heat or to cool the current sensor without the peripheries. The modulation and supply hardware should not be affected by the temperature. A design demand is that the chip and hardware have to be separated. For this reason a measurement board has to be developed. This hardware enables a very low signal damping and also guarantees good signals with defined edges.

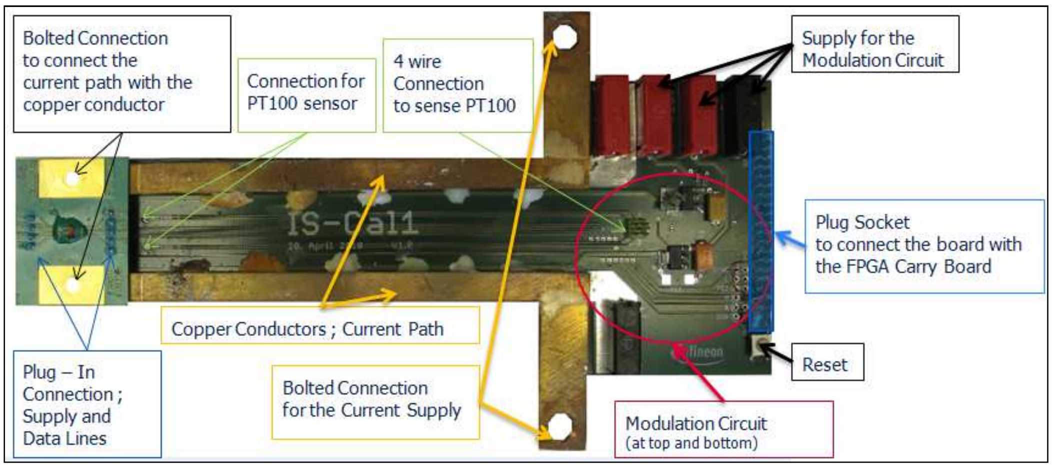

Bolted Connection to connect the current path with the copper conductor
Connection for PT100 sensor
4 wire Connection to sense PT100
Supply for the Modulation Circuit
Plug Socket to connect the board with the FPGA Carry Board
Copper Conductors ; Current Path
Reset
Plug – In Connection ; Supply and Data Lines
Bolted Connection for the Current Supply
Modulation Circuit (at top and bottom)
IS-Cal1

Figure 9.28 : Hardware for characterization of the CLMCS

Figure 9.28 shows the hardware with the copper conductor and a robust connection to ensure a small contact resistance. At the very right side of Figure 9.28 the plug socket to connect the IS-CAL1 – finger with the carrier board is highlighted in blue. The red circuit marks the area of the modulation circuit. Also the supply for the chip and the circuit are located in this area of the hardware. There are two massive copper conductors for the current supply. The connection between the supply lines and the copper conductor will be bolted at the two nine mm drills in the copper bars. On the other side of the measurement board is the plug-in connection for the current sensor. The chip will be supplied via these connections. Also the datelines will be connected via this plug- in connection. The connection between the copper conductors and the current rail from the CLMCS has to be established via a bolted connection. Therefore the six mm drills are in the copper conductor as well as in the CLMCS.

The carrier board also has to be used to connect the measurement hardware to the micro module. Figure 9.29 shows how the IS-CAL has to be connected to the TE0146-00 basis board.

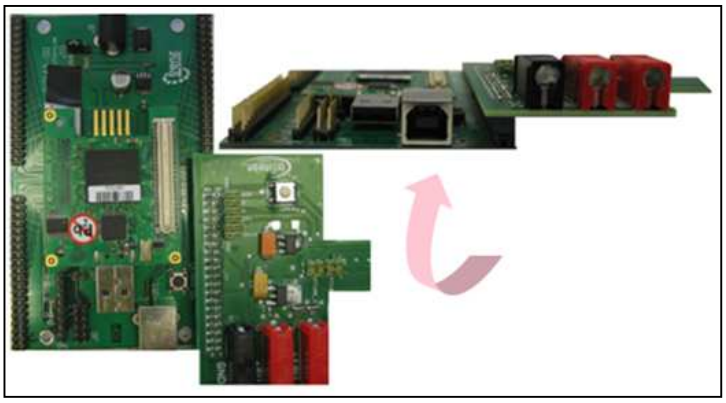


Figure 9.29 : Calibration – Finger connected with the Carrier Board

9.5.1 Hardware for the chip calibration

To allow measurements over a wide temperature range a thermo stream has been used for heating and annealing. Thus the chip has to be placed in a temperature chamber to enable a stable and homogenous ambient temperature. The size of the temperature chamber has been adapted to the inner diameter of the Helmholtz Coil. It is important that the current sensor is placed in the middle of the Helmholtz coil inside the temperature chamber to guarantee a homogenous field at the sensor. Figure 9.30 shows the setup to measure the magnetic field, inside the temperature chamber which has been produced by the Helmholtz coil.

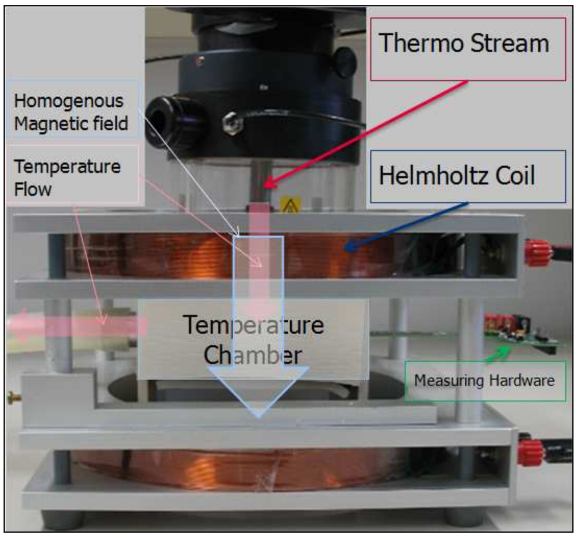


Figure 9.30 : Setup to measure the B field in a defined temperature range

Figure 9.31 shows the top view of the temperature chamber inside the Helmholtz coil.

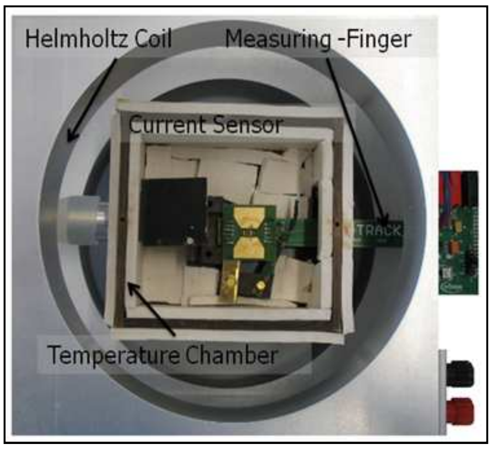


Figure 9.31 : Top view of the temperature chamber in the Helmholtz Coil

A small window at the front panel of the temperature chamber allows inserting the measurement finger with the current sensor.

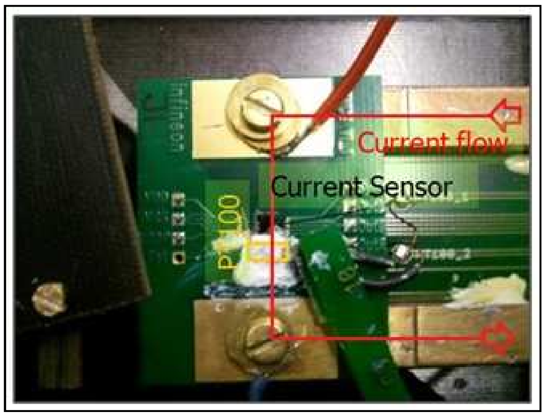


Figure 9.32 : CLMCS on copper conductor in the temperature chamber

Figure 9.32 shows how the PT100 sensor has been applied to the current sensor inside the temperature chamber. Heat transfer paste has been used to enable a good heat conductance between the PCB and the temperature sensor.

10 Software

The development of the RTL code for the FPGA has been done in VHDL with the Modelsim design suite. Before the code was synthesized the VHDL code was simulated with the software tool Modelsim. In order to simulate the current sensor behavioral model in conjunction with the FPGA code, the developed RTL model of the current sensor was provided by the digital designer team. The first communication from host PC to the FPGA was realized via the RS232 interface. Initially, a simple Terminal program was used to demonstrate the results. Finally the software MATLAB was used for the communication, the setup of the current sensor parameters and for all further calculations and representations. As mentioned the USB interface was used as final solution for the communication.

10.1 Data Processing

Figure 10.1 gives an overview of the data flow from the current sensor to the PC where the calculated current will be represented in a diagram on screen.

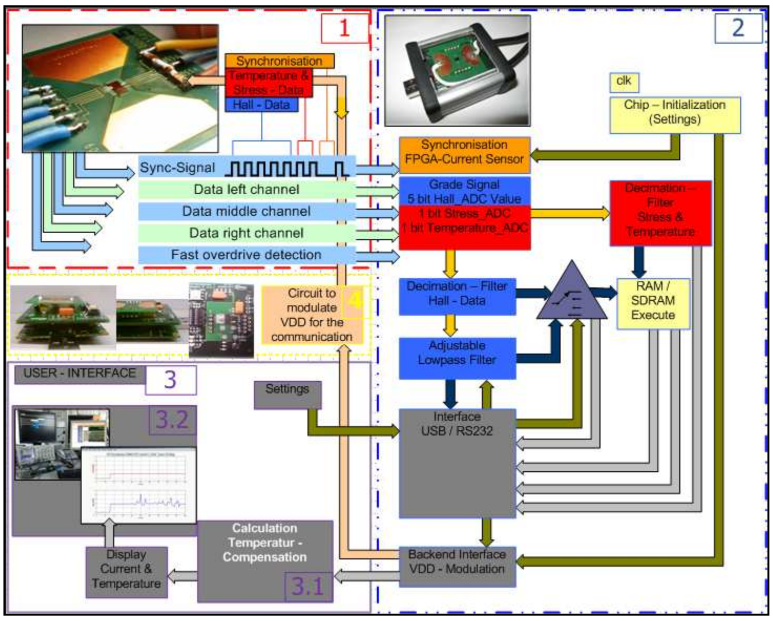

1
Synchronisation
Temperature & Stress - Data
Hall - Data
Sync-Signal
Data left channel
Data middle channel
Data right channel
Fast overdrive detection
2
clk
Chip – Initialization (Settings)
Synchronisation FPGA-Current Sensor
Grade Signal
5 bit Hall_ADC Value
1 bit Stress_ADC
1 bit Temperature_ADC
Decimation – Filter Stress & Temperature
Decimation – Filter Hall - Data
RAM / SDRAM Execute
Adjustable Lowpass Filter
Circuit to modulate VDD for the communication
4
USER - INTERFACE
3
Settings
3.2
Interface USB / RS232
Display Current & Temperature
Calculation Temperatur - Compensation
3.1
Backend Interface VDD - Modulation

Figure 10.1 : Data Process Overview and software concept of the Demonstrator tool

The top left hand sight of Figure 10.1 shows the current sensor and its PCB. The blue and green wires are connected to the data pins. The yellow and red wires are for the supply. As demonstrated in the figure, the sensor sends the data via the three channels as well as a via an additional signal, which indicates if a current overflow has been detected by the chip. The right part of the figure indicates the FPGA with the implemented instances for the signal processing. The yellow arrows show the raw data stream of the Hall, temperature and stress values. After processing of this data by the decimation filter and integration stages the data flow is indicated in blue. The green indicators show the instances where settings can be adjusted by the user while the data process is running. Examples of settings that can be adjusted include changing the cut off frequency of the low pass filters or sending the commands to modulate the backend interface of the chip. The possibility to buffer the processed data in the SDRAM before sending them to the PC can also be enabled. The left lower part of Figure 10.1 represents the user interface, respectively the PC where the data will be represented. This part allows controlling the FPGA by sending the corresponding commands to the hardware. The scaling from FPGA-filtered and processed raw data of the sensor to a readable format (i.e. amperes) is done in MATLAB by the host computer. The fourth part shown in the middle left hand side of the figure illustrates the modulation hardware. This circuit is connected with the FPGA and will be controlled by the backend interface instance of the VHDL code. The so called modulation circuit will be used to modulate the serial commands to the CLMCS to adjust the corresponding register settings within the current sensor digital part.

10.2 FPGA

10.2.1 User Interface

There are diverse possibilities to communicate with the demonstrator tool. The following explanation describes the functionality of the user interface. The demonstrator can either be controlled by the hardware interaction (switches, leds) or via a serial communication program in the MATLAB development environment. There is also a dip switch on the hardware which provides an option for making some settings or addressing of different devices in the future. LEDs are placed on the hardware to signal if the USB communication is working properly and if the communication between the chip and the hardware is stable. The second method for communication with the hardware is an interface based on a serial protocol via USB. Therefore a USB-Phy is implemented on the micro module which converts the serial protocol to the USB protocol. This communication allows data requests to be sent to the hardware or the data process on the FPGA to be changed.

Because the serial interface is based on a serial protocol like the RS232 protocol, it is only possible to send a bit stream with a data length of eight bit. However one command for the backend interface of the chip consists of twelve data bits as indicated in Figure 10.2. Therefore two eight bit commands from the PC have to be sent to the user interface on the FPGA via USB. The top instance 'testchip_t2' on the FPGA concatenates the two 8 bit commands to a 12 bit command after receiving the two bytes. Figure 10.2 shows how the commands are merged together. Furthermore figure 8.2 shows the format of the backend interface commands.

The left side of the figure represents how to interpret the data format on the PC before sending them to the FPGA. The highlighted red characters indicate the prefix of the twelve bit command. Table 10.12 shows the commands for the backend interface. Dependent on the command settings the most significant bits will be used to indicate the prefix of the command. Therefore, the first byte which will be sent to the FPGA represents the four bit head of the command. The second byte has to be including the remaining data. The so called data bits are highlighted in blue in Figure 10.2.

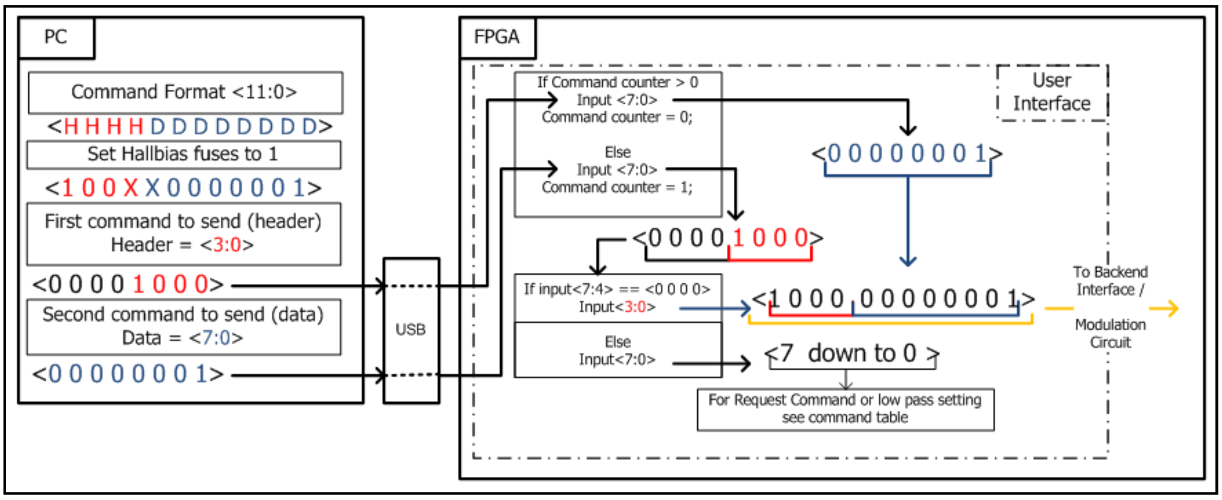


Figure 10.2 : User Interface – Command Format and Command Concatenating

If the upper side of the twelve bit command, which is represented by the four most significant bits, will be extended to four zero, the signal can then be separated into two eight bit signals. Assuming that a command consists of two eight bit signals where the four most significant bits are set to zero, the value of the upper nibble does not allow a value higher than fifteen in decimal. This limitation of the value range from the first nibble enables using the commands from the PC in order to control the filter settings or to choose in which format the FPGA responds by sending data to the PC. The command counter signal at the user interface on the FPGA toggles between zero and one every time the FPGA receives a command from the interface. At startup or after pressing the reset button the command counter is set to zero. Sending a decimal command with a value of fifteen twice from the PC to the FPGA will cause an initialization on the FPGA and will set the command counter to zero. Figure 10.2 shows how the commands will be handled dependent on the state of the command counter. If the command counter is zero and no command has been sent to the FPGA, the user interface will check the upper four bits from the first command received. The data format on the PC and on the FPGA is identical. That means, by sending the value fifteen in decimal from the PC via a terminal program in an unsigned integer format, the four upper bits from the received commands are zero and the least significant bits are one at the FPGA. Figure 8.2 shows, that if the four most significant bits of the first command are zero and the command counters state is also zero, the four least significant bits of the sent signal will be transferred into the four most significant bits of the command register. This register represents the command for the backend interface. Now the command counter will be set to one and the user interface will concatenate the next eight bit command from the interface with the previous saved bits in the command register at the lower side of the three nibbles.

This twelve bit command will be transferred to the backend interface. At this time the command counter will be set to zero and the interface will check the upper four bytes from the next command which will be sent to the FPGA again.

It is also possible to setup the FPGA in a way that it only sends the five bit Hall data with no advance signal processing.

If the user would like to send a command for the backend interface, the value of the first nibble has to be smaller than sixteen. That means that the upper four bits of the initial command has to be zero. This indicates that the lower four bits are used for the upper side of the backend interface command. After receiving the second byte the concatenated command is given to the backend interface and will be modulated by the chip.

10.2.2 Commands for data request

To control the interface on the FPGA in conjunction with the current sensor the computer program MATLAB has for the most part been used. Of course any basic terminal program which allows controlling the com ports on the PC can be used for communication. In this thesis communication using the MATLAB tool will be described. The following command line has to be used to connect to the serial port in MATLAB:

Fileref =serial('comx','baudrate',921600,'stopbit',1);

Dependent on the quantity of expected data, the input buffer size has to be set accordingly, by using the next command line.

Fileref.inbutpuffersize = 2e5; % setting the input buffer size to the value $2 \cdot 10^5$.

Before any request to the FPGA can be sent, the com board has to be opened with the 'fopen'-command. Table 10.1 too Table 10.4 include all commands as well as the sequence on how the data will be sent to the PC. As described, it is possible to initialize the FPGA by sending the command for the software reset. Therefore the decimal value fifteen has to be sent twice to the hardware.

The following paragraph describes how to send a command with the software MATLAB. The code for sending a command begins with the '*fwrite*'-instruction. '*Fwrite*' is a function that allows sending a value to the serial interface. The reference to the serial port, the value to send and also the format how the value has to be sent are the requested inputs of the '*fwrite*'- function. In the following example the decimal value fifteen will be send to the serial interface, which has been referenced to the 'fileref'-variable. Either the binary value '00001111' with the '*bin2dec*'-function or the decimal value fifteen has to be passed to the '*fwrite*'-function. At this point it is important that the format is always set to 'unsigned integer 8', which will be represented by the 'uint8'-string inside the brackets from the '*fwrite*'-function call.

fwrite(fileref,bin2dec('00001111'),'uint8'); % sending the value 15 to the uart *board.fwrite(fileref,15, 'uint8','async');* % sending the value 15 to the uart board.

Request command from the PC to the user interface	
Range of the command in decimal: 240-255	
Request from the FPGA	
Bytes	Data
1	Don’t care
2	“000” & 5bit raw Hall data left
3	“000” & 5bit raw Hall data middle
4	“000” & 5bit raw Hall data right
5	<23 down to 16> Hall left
6	<15 down to 8> Hall left
7	<23 down to 16> Hall middle
8	<15 down to 8> Hall middle
9	<23 down to 16> Hall right
10	<15 down to 8> Hall right
11	“000” & 5bit raw Hall data left
12	“000” & 5bit raw Hall data middle
13	“000” & 5bit raw Hall data right
14	“0000”& <11 down to 8>
15	<7 down to 0> stress data middle
16	Decimal 71
17	“0000” & <11 down to 8>
18	<7 down to 0> stress data left
19	“0000” & <11 down to 8>
20	<7 down to 0> temperature data right
21	Decimal 72
22	“0000” & <11 down to 8>
23	<7 down to 0> temperature data middle
24	“0000” & <11 down to 8>
25	<7 down to 0> temperature data left
26	Decimal 10

Table 10.1 : Corresponding data format from the FPGA to the request

Range of the command in decimal: 224 – 239	
Request from the FPGA	
Byte	Data
1	Decimal 71
2	<23 down to 16> Hall left
3	<15 down to 8> Hall left
4	<7 down to 0> Hall left
5	<23 down to 16> Hall middle
6	<15 down to 8> Hall middle
7	<7 down to 0> Hall middle
8	<23 down to 16> Hall right
9	<15 down to 8> Hall right
10	<7 down to 0> Hall right
11	Decimal 10

Table 10.2 : Corresponding data format from the FPGA to the request values 224-239

Request command from the PC to the user interface	
Range of the command in decimal: 80 - 95	
Request from the FPGA	
Bytes	Data
1	Decimal 14
2	Decimal 14
3	"0000" & 11 down to 8 temperature left
4	7 down to 0 temperature left
5	15 down to 8 Hall left
6	7 down to 0 Hall left
7	Dec 14
8	Dec 14
9	"0000" & 11 down to 8 temperature middle
10	7 down to 0 temperature middle
11	15 down to 8 Hall middle
12	7 down to 0 Hall middle
13	Decimal 14
14	Decimal 14
15	"0000" & 11 down to 8 temperature right
16	7 down to 0 temperature right
17	15 down to 8 Hall right
18	7 down to 0 Hall right
19	Dec 14
20	Dec 14
21	"0000" & 11 down to 8 stress middle
22	7 down to 0 stress middle
23	"0000" & 11 down to 8 stress left
24	7 down to 0 stress left
25	Decimal 14
26	Decimal 14
27	15 down to 8 time step
28	7 down to 0 time step
29	"0000" & 11 down to 8 stress right
30	7 down to 0 stress right

Table 10.3 : Corresponding data format from the FPGA to the request values 80-95

Request command from the PC to the user interface	
Range of the command in decimal: 112 - 127	
Request from the FPGA	
1	Decimal 14
2	Decimal 14
3	"000" & 4 down to 0 Hall raw left
4	"000" & 4 down to 0 Hall raw middle
5	"000" & 4 down to 0 Hall raw middle
6	Don't care

Table 10.4 : Corresponding data format from the FPGA to the request values 112-127

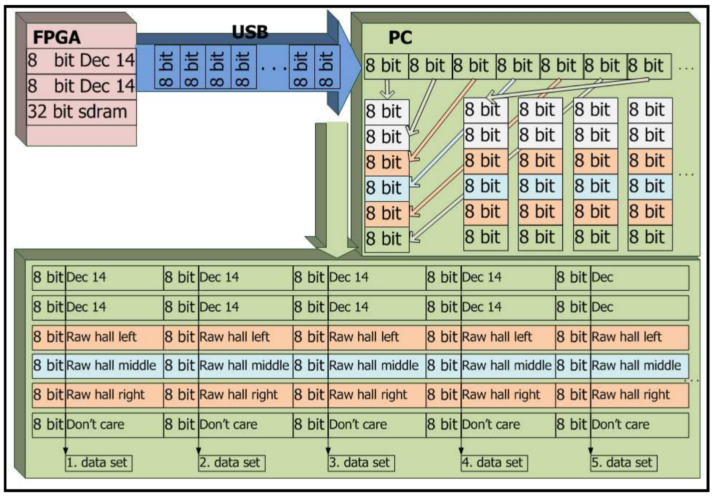


Figure 10.3 : Data Protocol of the buffered 5bit raw Hall data from the FPGA to the PC

Figure 10.3 and Figure 10.4 show the data format sent by the FPGA to the PC, as a result of the initial request command from the PC. If the request command 112_{10} is sent to the FPGA, the device responds in a data format as shown in Table 10.4. Figure 10.3 shows how the data has to be interpreted to get valid data sets. The first two bytes are used as control bytes. The value 14_{10} indicates that a new data set will begin. The last byte will not be used. Because of the transmission protocol shown in Figure 10.3, one sub data set consists of two bytes with the value 14_{10} and four bytes from the CLMCS conditioned from the FPGA. If the user request command 80_{10} is sent, the FPGA responds as described in Table 10.3. Figure 10.4 shows that one data set consists of five sub data sets if the value of the rewritten command is 80_{10}. For noise measurements the raw data from the Hall plates are needed in order to calculate the Fourier transformation which delivers information about the behavior of the system in the frequency domain. In this case, only the five bit raw data from the Hall plates are important, whereas all data from the CLMCS are needed for the current calculation. In order to receive all buffered processed data from the FPGA, the command 80_{10} has to be sent to the hardware. The first two bytes of this data set always start with the value 14_{10}. This indicates that a new data set will begin. In the case where the command 80_{10} is used, all decimated and filtered values from the FPGA are transmitted to the PC. Because of the higher information content of the data, the number of bytes increases. Therefore five times six bytes has to be sent for one data set. The data starts with the same value as explained with the amount 14_{10} at position one and two, followed by 32 bytes from the FPGA as described in Table 10.3.

10.2.3 Commands to set the IIR low pass filter

The user interface can also be used to setup the cutoff frequency of the two implemented IIR low pass filters. It is also possible to cascade the output of the first filter with the input of the second filter by sending the corresponding command. The command counter value has to be zero, which either means no command has yet been received or an even number of commands have been received from the FPGA. The command to setup the low pass settings has to begin with "110". If the three most significant bits from the received command has the value six, or the bits from seven down to five are set to "110" in binary, then the user interface will use the remaining five bit value for the filter setup. Table 10.6 shows the five bit setup value in the setup value column and the corresponding cutoff frequency on the same line.

After the startup sequence from the FPGA or following a reset command, either by sending the command value fifteen in decimal twice via the interface or by pressing the reset button on the hardware, the multiplication registers 'adjust_s' and 'a1_v' for both low pass filters are set to zero. This prevents low pass filtering of the Hall data after the decimation filter from occurring. By sending the command value 200 in decimal to the user interface, the second IIR filter will be cascaded to the first low pass, and the cutoff frequency of the second low pass will be set to the value of the initial filter. This command corresponds with the setup value eight from Table 10.6, because the prefix '110' and the setup value '01000' are representing the value 200 in decimal. By sending the value 223 in decimal the second filter is set to 'feed through' and the Hall data will only be filtered by a first order low pass structure.

Setup value	K (sum of shifts)	Cutoff frequency in Hz		Setup value	K (sum of shifts)	Cutoff frequency in Hz
				14	13/32	396
27	13/16	1032		13	3/8	360
26	25/32	968		12	11/32	294
25	3/4	907		11	5/16	262
24	23/32	849		10	9/32	232
23	11/16	793		9	1/4	202
22	21/32	741		7	7/32	174
21	5/8	691		6	3/16	146
20	19/32	643		5	5/32	120
19	9/16	597		4	1/8	94
18	17/32	554		3	3/32	70
17	1/2	512		2	1/16	46
16	15/32	472		1	1/32	22
15	7/16	433		0	0	Feedthrough

Table 10.5 : Setup Values to set the cutoff frequency of the IIR filter

Table 10.6 : Settings for the IIR low pass filters

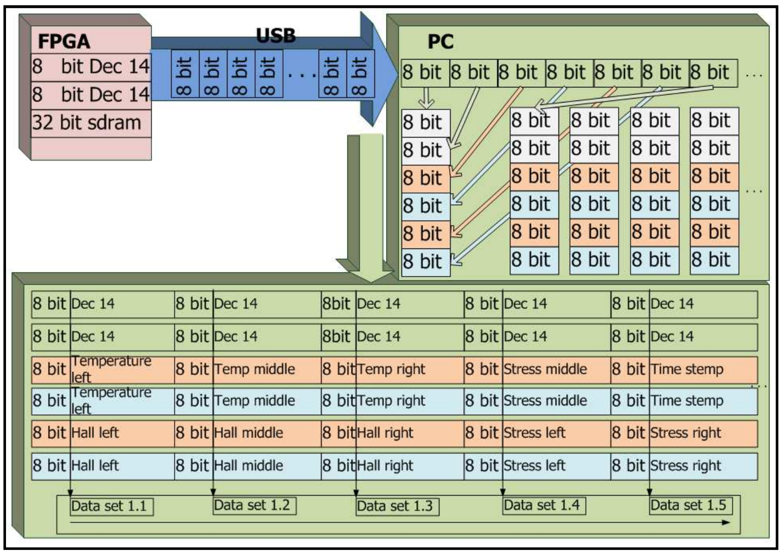


Figure 10.4 : Data Protocol of all processed data from the FPGA to the PC

10.2.4 SDRAM

As previously mentioned the chip consists of three channels. They are all working in parallel and sending data at the same time to the FPGA. All computations which are necessary to calculate the measured data from the Hall plates into an adequate current are outsourced to an external computer. To ensure a time consistent calculation on the PC, which gets the data with a delay from the sensor, dependent on the serial interface between the FPGA and the PC, it is necessary to buffer all data before sending them to the PC. To allow an alternating current analysis or filtering of the data after the sending process, a large amount of data is required for a Fourier algorithm or moving average filters on the PC. Therefore a SDRAM on the FPGA board is used to buffer the data from the current sensor.

The following settings and protocols have to be realized to ensure correct communication between the FPGA and the SDRAM.

10.2.5 SDRAM overview

In this project a synchronous DRAM MT48LC2M32B2 – 512K x 32 x 4 from the company micron was used to buffer all the data as well as intermediate the results. The memory size of the module allows savings of 64M bytes. The storage is structured to consist of four separated banks. Each of them consists of 2048 rows and 256 columns in which every cell has a capacity of 32 bits. Figure 10.5 shows how the SDRAM is structured.

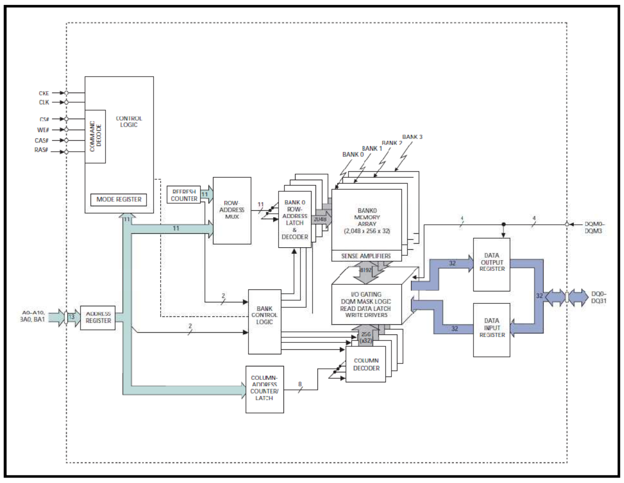


Figure 10.5 : Block diagram of the SDRAM [7]

10.2.6 Instance mySDRAM

The instance 'mysdram' manages the data process between the FPGA and the SDRAM. The micron data sheet describes in detail how to initialize the memory. The protocol and timing specifications for the write and read commands are also described in detail on the data sheet. The netlist in the appendix shows how the control, address and data wires are connected with the FPGA. Dependent on the data request command from the user interface, the input register of the 'mysdram'-instance is loaded with the five bit raw data from the Hall plates or with the sixteen bit filtered Hall data from the decimation filter. If the filtered values are to be saved, the twelve bit temperature and the twelve bit stress data will also be saved and send to the PC.

Dependent on the sent command, the FPGA regulates the format in which the data will be shared. For noise measurements it is possible to save only the five bit data streams from the three Hall channels. Figure 10.6 shows how the five bit raw data are stored in the SDRAM.

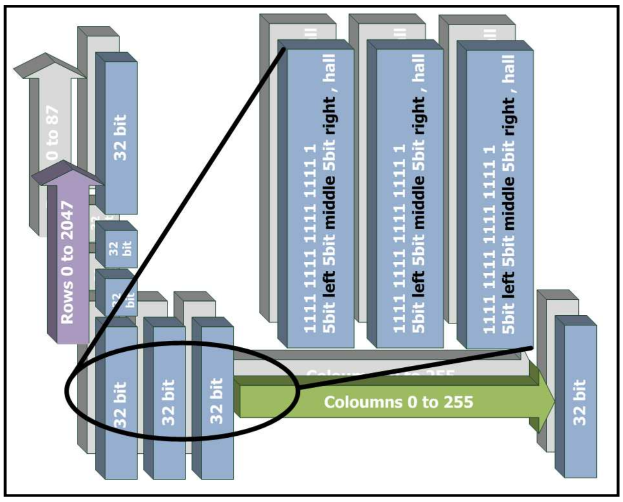


Figure 10.6 : Saving Format of the 5 bit raw data in the SDRAM

To get all information from the chip, the command 80_{10} has to be sent to the user interface via USB. After receiving the command, the decimated Hall signals will be buffered into the SDRAM and also the temperature and stress values are available in the SDRAM. Figure 10.7 shows how the whole processed data from the CLMCS will be stored in the SDRAM.

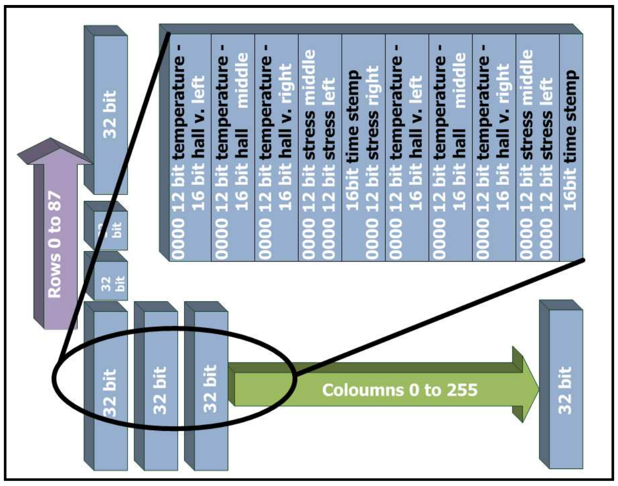

32 bit
Rows 0 to 87
32 bit
0000 12 bit temperature - 16 bit hall v. left
0000 12 bit temperature - 16 bit hall middle
0000 12 bit temperature - 16 bit hall v. right
0000 12 bit stress middle
0000 12 bit stress left
16bit time stemp
0000 12 bit stress right
0000 12 bit temperature - 16 bit hall v. left
0000 12 bit temperature - 16 bit hall middle
0000 12 bit temperature - 16 bit hall v. right
0000 12 bit stress middle
0000 12 bit stress left
16bit time stemp
32 bit
32 bit
32 bit
Coloumns 0 to 255
32 bit

Figure 10.7 : Saving Format of all processed data from the CLMCS in the SDRAM

The 'mysdram'-instance is structured into four main sequences. At the beginning an initialization has to be done. Afterwards the write and read processes will be executed alternately. Every 15.6 µs after the initialization step, a refresh has to be done to ensure data integrity. Every command is available for one clock cycle to ensure that the command will be transferred to the SDRAM with the rising edge of the clock.

10.2.7 Initialization

After the 100µs power up period, the clock enable signal has to be set to high and the initialization starts. All required time steps in the write, read, refresh and initialization protocol depend on the time period 't_cke' from the clock. Table 10.7 shows all relevant time steps which are used in the respective protocols.

Relevant time values to communicate with the SDRAM			
Parameter	Symbol	Value	Unit
Clock cycle time	t_{ck}	33.3	Ns
Precharge command	t_{RP}	100	Ns
Auto Refresh Period	t_{RFC}	333	Ns
Load Mode Register	t_{MRD}	100	Ns
Active to read or	t_{RCD}	100	Ns
Active to Precharge	t_{RAS}	233	Ns
Precharge command	t_{RP}	100	Ns
Active to Active	t_{RC}	600	Ns

Table 10.7 : Relevant time steps for the SDRAM

A more detailed time description can be found on the micron data sheet. In addition the power on clock enable signal has to set to zero for at least 100µs. In this period the supply as well as the clock signal of the SDRAM has to be stable. Figure 10.8 shows the protocol for the initialization mode.

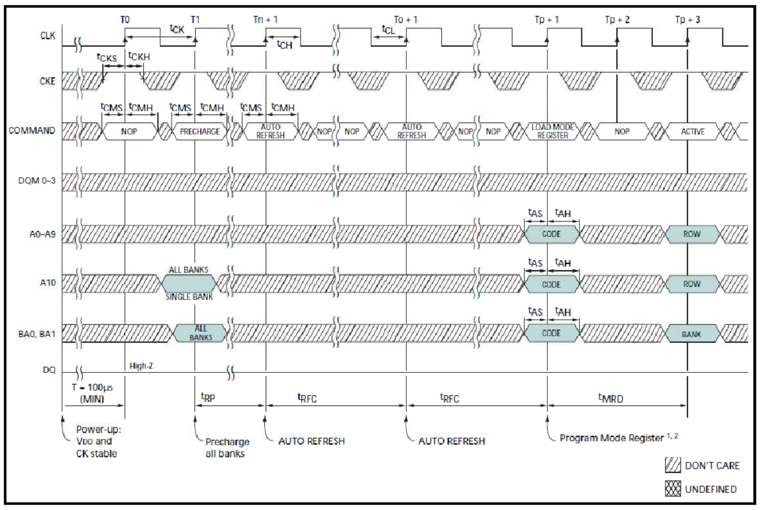


Figure 10.8 : Protocol for the SDRAM initialization [7]

The clock signal ‘cke’ will be derived from a 60MHz system clock. Therefore the system clock has to be divided by two. Figure 10.9 shows a RTL level simulation of the system clock and the derived clock. Furthermore, the initialization process is represented.

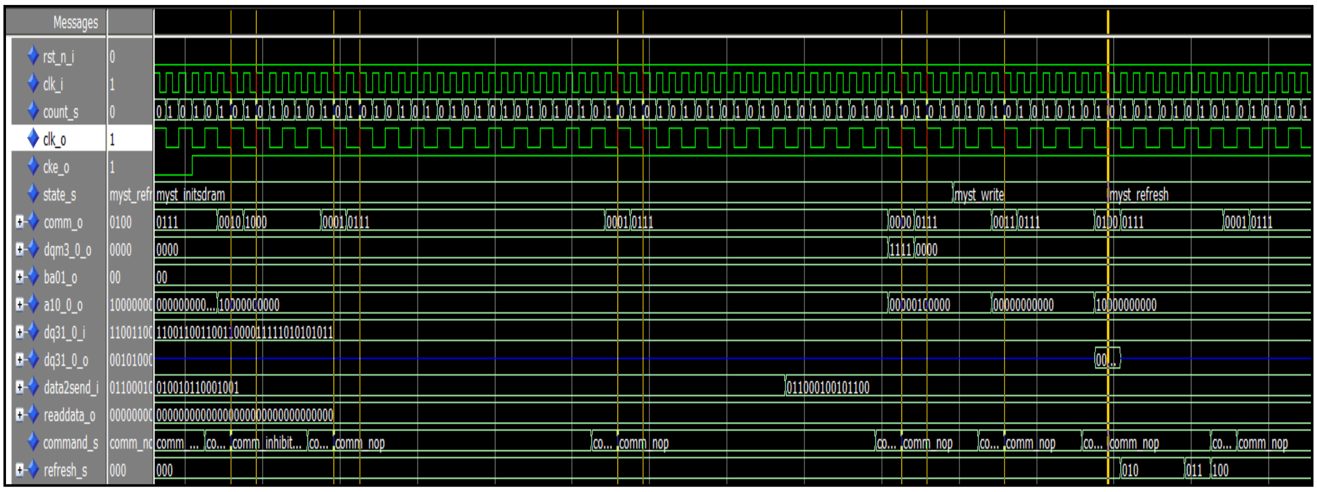


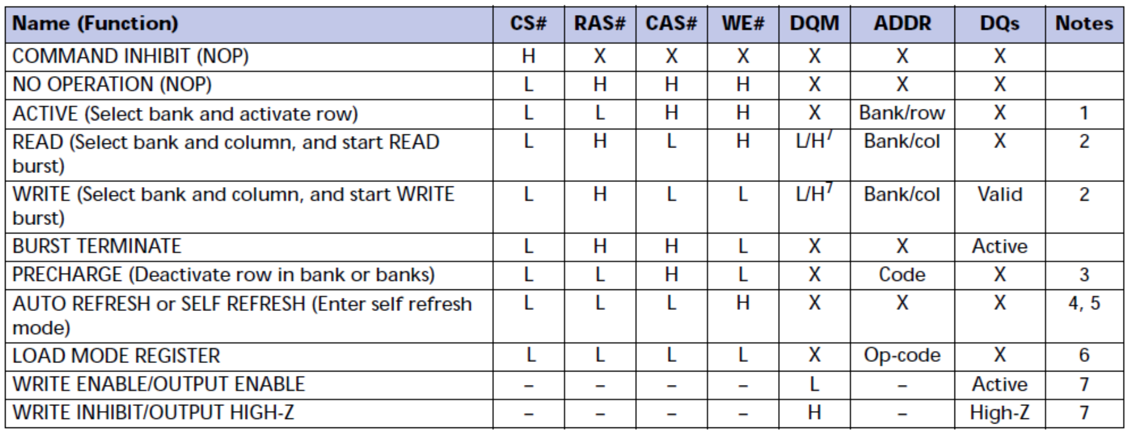
Figure 10.9 : Write simulation of the SDRAM with MODELSIM

Figure 10.9 shows a writing simulation of the SDRAM after the initialization. In the top the system clock ‘clk_i’ can be seen. The ‘clk_o’ signal represents the derivative clock for the SDRAM. Below the clock signal the cke_o signal is set to high to enable the initialization. The state_s signal indicates the actual state of the process flow. After the ‘state_s’ signal changes from the ‘myst_init’ into the ‘myst_write’ status the initialization has been done. Figure 10.9

represents all commands, respectively states and address signals which have to be set for the initialization and write state. The commands will be enabled with every rising clock edge of the ‘clk_o’-signal. The yellow coursers indicate the sampling time where the commands will be enabled.

After a 100µs pause, the ‘cke_o’ signal has to be set to a high state. At this time a ‘nop’–command has to be sent with the rising clock edge from the ‘clk_o’ signal. This command is followed by the ‘precharge’ command with the next rising clock edge. After this command the ‘inhibit’ command has to be sent for three clock cycles to bridge the time gap ‘t_{RP}’. Afterwards the ‘auto refresh’ command has to be sent for the first time. Now the ‘nop’ command has to be sent ten times synchronously with every clock cycle. This time duration is called the ‘t_{RFC}‘ period. After this period the second ‘auto refresh’ command has to be sent. Afterwards the ‘nop’ command has to be sent ten times again before sending the ‘load mode register’ command by the next rising clock edge. After sending this command, the mode register has to be set. Furthermore, the command ‘load mode register’ has to be applied on the control signals and at the same time the register will be loaded with the address signals 10 down to 0 as it is shown in Figure 10.9. After the setup from the mode register, the initialization will be finished by sending the ‘nop’ command at least three times, to temporize the ‘t_{mrd}‘-cycle.

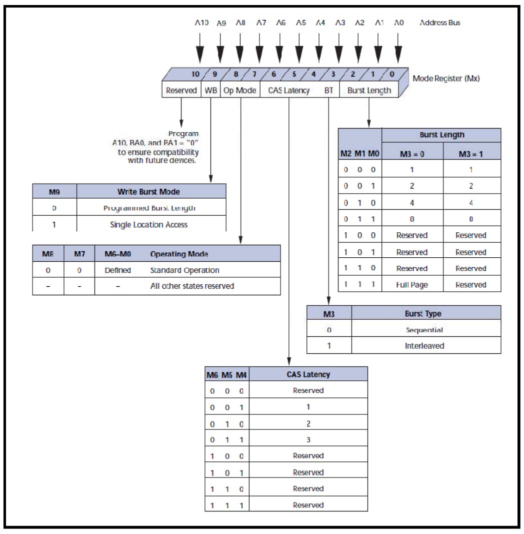
Table 10.8 shows how the input signals from the control logic of the SDRAM have to be set for the particular commands.



Name (Function)	CS#	RAS#	CAS#	WE#	DQM	ADDR	DQs	Notes
COMMAND INHIBIT (NOP)	H	X	X	X	X	X	X	
NO OPERATION (NOP)	L	H	H	H	X	X	X	
ACTIVE (Select bank and activate row)	L	L	H	H	X	Bank/row	X	1
READ (Select bank and column, and start READ burst)	L	H	L	H	L/H[7]	Bank/col	X	2
WRITE (Select bank and column, and start WRITE burst)	L	H	L	L	L/H[7]	Bank/col	Valid	2
BURST TERMINATE	L	H	H	L	X	X	Active	
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	X	Code	X	3
AUTO REFRESH or SELF REFRESH (Enter self refresh mode)	L	L	L	H	X	X	X	4, 5
LOAD MODE REGISTER	L	L	L	L	X	Op-code	X	6
WRITE ENABLE/OUTPUT ENABLE	–	–	–	–	L	–	Active	7
WRITE INHIBIT/OUTPUT HIGH-Z	–	–	–	–	H	–	High-Z	7

Table 10.8 : Command setup for the SDRAM [7]

Figure 10.10 shows the structure of the load mode register.


A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 Address Bus
10 9 8 7 6 5 4 3 2 1 0
Mode Register (Mx)
Reserved
WB
Op Mode
CAS Latency
BT
Burst Length
Program A10, BA0, and BA1 = "0" to ensure compatibility with future devices.

M9	Write Burst Mode
0	Programmed Burst Length
1	Single Location Access

M8	M7	M6–M0	Operating Mode
0	0	Defined	Standard Operation
–	–	–	All other states reserved

M2 M1 M0	Burst Length	
	M3 = 0	M3 = 1
0 0 0	1	1
0 0 1	2	2
0 1 0	4	4
0 1 1	8	8
1 0 0	Reserved	Reserved
1 0 1	Reserved	Reserved
1 1 0	Reserved	Reserved
1 1 1	Full Page	Reserved

M3	Burst Type
0	Sequential
1	Interleaved

M6 M5 M4	CAS Latency
0 0 0	Reserved
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	Reserved
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

Figure 10.10 : Load Mode Register of the SDRAM [7]

Table 10.9 shows the adjusted values in the load mode register.

Description	**Value**
Burst Length	One
Burst Type	Sequential
CAS Latency	Three
Operating Mode	Standard Operation
Write Burst Mode	Programmed Burst Length

Table 10.9 : Setup for the load mode register of the SDRAM [7]

10.2.8 Write state

The protocol in Figure 10.11 shows that every write procedure starts with an activate command followed by the 'nop' commands.

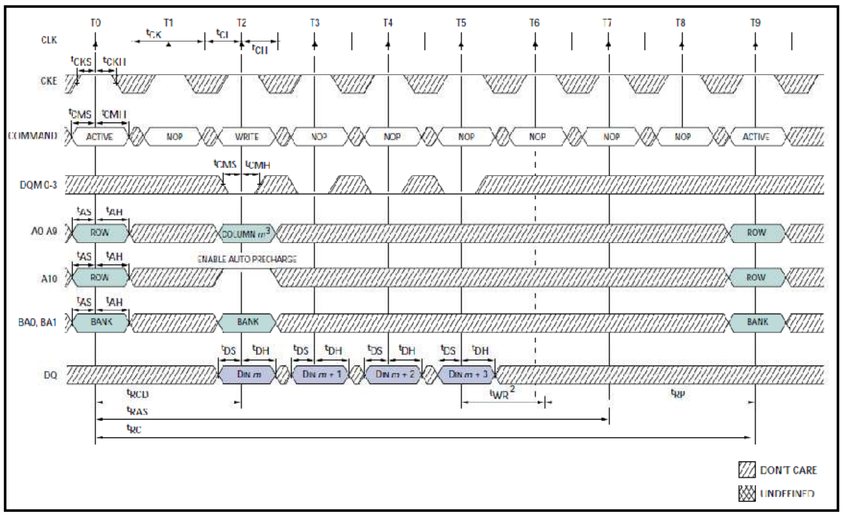


Figure 10.11 : Write Protocol of the SDRAM [7]

To enable the 'auto precharge' mode, the highest bit of the address bus has to be set to high, when sending the write command. At this time the data which is to be saved in the SDRAM has to be transferred to the data bus. The bits from nine down to zero of the address bus include the column address at this time. The DQM bits are always set to zero while the FPGA is working in the write state. The row address has to be set synchronously with the address bits with the 'activate' command at the beginning of a write cycle. At this time point the bank address has to be set with the ba_01 bits. After each write cycle the address counter will be increased by one. That means the column counter will be increased by one. If the address of the column counter reaches the value 255, the column counter will be set back to zero and the row counter will be increased by one. And if the row counter will be equal to the value 2047, the bank address will be increased by one. The FPGA changes into the reading state if the adjusted bank address, as well as the row address are exceeded. Depending on if the FPGA saves the five bit raw data, or if all filtered and decimated values are saved, the adjusted addresses are different. By sending the request command with the value 80 in decimal to the user interface, the SDRAM will set the address limit as follows. The bank address limit will be adjusted to one and the row address will be set to 253. In this case only the five bit raw data from the three Hall channels will be saved. If the command with the decimal value 80 is sent to the FPGA, the limit for the bank address is set to zero and the upper limit of the row address is set to 86. In this case all filtered and decimated values from the three channels are buffered. If the row and the bank address have reached the limit, the FPGA will switch to the reading mode and sets all address counters to zero. After each write cycle a refresh cycle will be done. As mentioned before at least every 15 µs, a refresh has to be done to avoid data loss. If the command 80 has been sent to the user interface every 0.21ms a new data set will be available to store the data into the SDRAM. In the case where the user command 80_{10} has been sent, one data set has to be split into five 32 bit blocks. Therefore the time cycle to save one data set could be calculated as follows. One write cycle lasts 533ns. After each write cycle one refresh cycle will be executed. Every refresh cycle lasts approximately 470µs.

Equation 10.1

describes how long one saving cycle for one data set lasts if the user command 80_{10} has been received from the user interface.

$$t\_save\_cycle = 5 * t\_write\_cy + 5 * t\_refresh\_cy$$

Equation 10.1

$$t\_save\_cycle = 5 \cdot 300ns + 5 \cdot 300ns = 3us$$

Equation 10.2

In the period where no new datasets are available for saving in the SDRAM, a refresh cycle will be made by the FPGA every 3µs. In the case where the user command 112_{10} has been sent to the user interface, the SDRAM will receive new raw data from the CLMCS every 1µs. In this case one data set consists of fifteen bits, which needs one writing cycle to save it in a column of the SDRAM, which has a capacity of 32 bits. Every write cycle will be followed by a refresh cycle. The time frame of the two cycles is smaller than the update rate of the raw data. One write cycle lasts 300ns and the refresh cycle also lasts approximately 300ns. The update rate depends on the output signal from the CLMCS. As previously mentioned, the CLMCS transfers a new data set from the Hall plates every 1µs.

Figure 10.9 shows how the simulation of a write cycle is implemented on the FPGA. After sending the activation command, three 'nop' commands will be sent before the write command is enabled. After sending the write command, four ‘nop’ commands will be sent to temporize the ‘t_{rp}‘ period and to finish the write cycle. When sending the write command, the data bus needs to include the valid data which has to be saved in the SDRAM. Otherwise the data bus has to be set to high z.

10.2.9 Read state

Figure 10.12 represents the reading protocol. Dependent on the latency settings which are represented in Table 10.10, the data sets to read are valid at the appointed time after sending the read command. The read cycle will start in the same way as the write period by sending the activation command. The address bit has to include the row address at this point and the b10 bits have to represent the bank address at the same time. In the next step the nop command will be sent to the SDRAM three times. Afterwards the read command has to be sent with the next rising edge of the SDRAM clock. At this time the DQM bits have to be zero and the address bits 9 down to 0 has to include the address of the column. The most significant bit from the address bus has to be set to high to enable the ‘auto refresh’ mode. Due to the CAS - latency setting which is set to three, the information on the data bus will be valid at the third ‘nop’ command after sending the ‘read’- command. The read cycle will be finished by sending one more ‘nop’ command. Therefore the read command needs the same time period as the ‘write’ command. In the same way as in the write state, a refresh will be done after each read cycle and the address will be increased by one. If the row, respectively the bank address reaches the upper limit, all address counters will be set to zero and the FPGA will change into the write mode again.

10.2.10 Refresh state

Figure 10.13 shows the protocol how the auto refresh has to be done. The refresh command has to be executed at least every 15 µs. In the FPGA the auto refresh will appear after every write and read cycle. If the FPGA is in the write mode and no new data is available to save into the SDRAM a refresh cycle will be done every 3 µs.

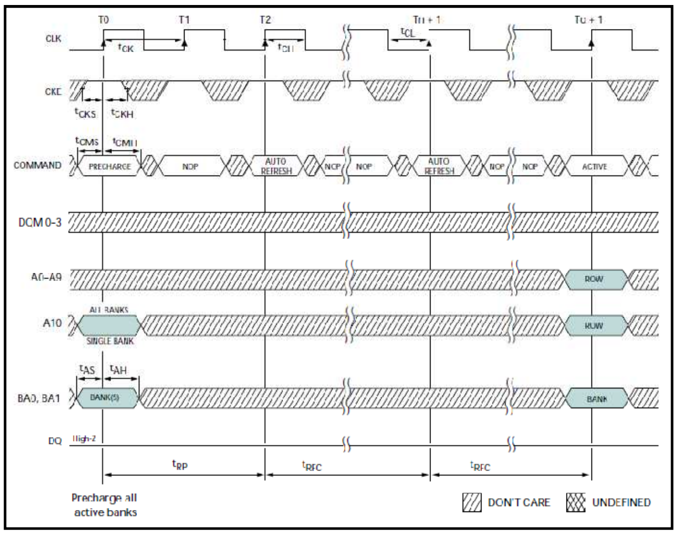


Figure 10.12 : Reading Protocol of the SDRAM [7]

Speed	Allowable Operating Frequency (MHz)		
	CL = 1	CL = 2	CL = 3
-5	–	–	≤ 200
-55	–	–	≤ 183
-6	≤ 50	≤ 100	≤ 166
-7	≤ 50	≤ 100	≤ 143

Table 10.10 : CAS Latency setting for the SDRAM [7]

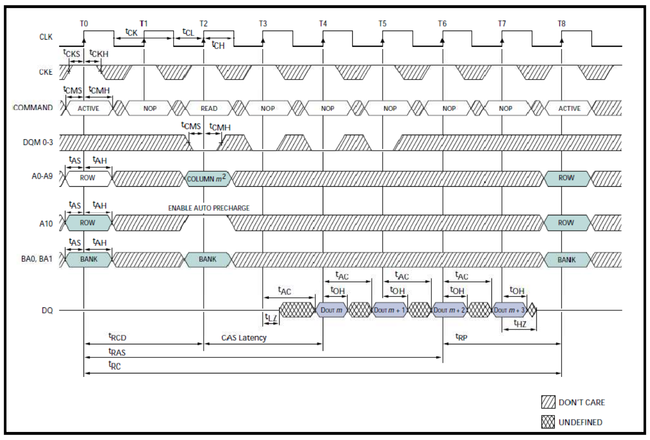


Figure 10.13 : Refresh Protocol of the SDRAM [7]

10.2.11 Instance Dut_current_top

This instance is used for the synchronization with the data stream from the current sensor. Figure 10.14 shows the Hall data format from the CLMCS.

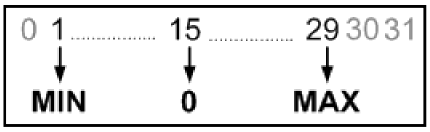
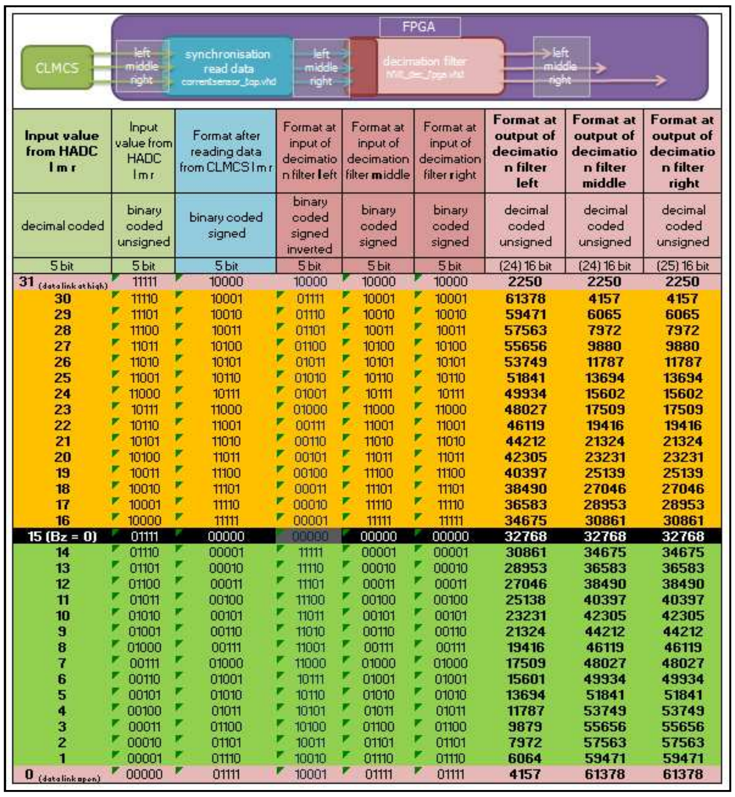


Figure 10.14 : Hall Data format from the CLMCS

A separate signal is used for the synchronization. There is an 8 MHz signal available at the sync pin of the chip which allows correct synchronization. The data gap at the 8th bit on this signal enables correct sampling of the three input data signals from the CLMCS on the FPGA. The data signals will be valid at each rising edge of the sync signal. The first five bits include the Hall data. The two least significant bits include the temperature and stress information.

Figure 9.6 shows how temperature and stress information is multiplexed. There are alternately 4096 bits representing the actual temperature value and 4096 bits which include the updated stress information from the current sensor. The least significant bit represents the stress value while the second bit includes the temperature information. The bit format is shown in Figure 9.8. To allow the decimation of the data, as it is described below, it is necessary to transform the data to a corresponding format as shown in Figure 10.15.


FPGA
CLMCS
left
middle
right
synchronisation
read data
currentsensor_top.vhd
left
middle
right
decimation filter
left
middle
right

Input value from HADC l m r	Input value from HADC l m r	Format after reading data from CLMCS l m r	Format at input of decimation filter left	Format at input of decimation filter middle	Format at input of decimation filter right	**Format at output of decimation filter left**	**Format at output of decimation filter middle**	**Format at output of decimation filter right**
decimal coded	binary coded unsigned	binary coded signed	binary coded signed inverted	binary coded signed	binary coded signed	decimal coded unsigned	decimal coded unsigned	decimal coded unsigned
5 bit	5 bit	5 bit	5 bit	5 bit	5 bit	(24) 16 bit	(24) 16 bit	(25) 16 bit
31 (data link at high)	11111	10000	10000	10000	10000	**2250**	**2250**	**2250**
30	11110	10001	01111	10001	10001	**61378**	**4157**	**4157**
29	11101	10010	01110	10010	10010	**59471**	**6065**	**6065**
28	11100	10011	01101	10011	10011	**57563**	**7972**	**7972**
27	11011	10100	01100	10100	10100	**55656**	**9880**	**9880**
26	11010	10101	01011	10101	10101	**53749**	**11787**	**11787**
25	11001	10110	01010	10110	10110	**51841**	**13694**	**13694**
24	11000	10111	01001	10111	10111	**49934**	**15602**	**15602**
23	10111	11000	01000	11000	11000	**48027**	**17509**	**17509**
22	10110	11001	00111	11001	11001	**46119**	**19416**	**19416**
21	10101	11010	00110	11010	11010	**44212**	**21324**	**21324**
20	10100	11011	00101	11011	11011	**42305**	**23231**	**23231**
19	10011	11100	00100	11100	11100	**40397**	**25139**	**25139**
18	10010	11101	00011	11101	11101	**38490**	**27046**	**27046**
17	10001	11110	00010	11110	11110	**36583**	**28953**	**28953**
16	10000	11111	00001	11111	11111	**34675**	**30861**	**30861**
15 (Bz = 0)	01111	00000	00000	00000	00000	**32768**	**32768**	**32768**
14	01110	00001	11111	00001	00001	**30861**	**34675**	**34675**
13	01101	00010	11110	00010	00010	**28953**	**36583**	**36583**
12	01100	00011	11101	00011	00011	**27046**	**38490**	**38490**
11	01011	00100	11100	00100	00100	**25138**	**40397**	**40397**
10	01010	00101	11011	00101	00101	**23231**	**42305**	**42305**
9	01001	00110	11010	00110	00110	**21324**	**44212**	**44212**
8	01000	00111	11001	00111	00111	**19416**	**46119**	**46119**
7	00111	01000	11000	01000	01000	**17509**	**48027**	**48027**
6	00110	01001	10111	01001	01001	**15601**	**49934**	**49934**
5	00101	01010	10110	01010	01010	**13694**	**51841**	**51841**
4	00100	01011	10101	01011	01011	**11787**	**53749**	**53749**
3	00011	01100	10100	01100	01100	**9879**	**55656**	**55656**
2	00010	01101	10011	01101	01101	**7972**	**57563**	**57563**
1	00001	01110	10010	01110	01110	**6064**	**59471**	**59471**
0 (data link open)	00000	01111	10001	01111	01111	**4157**	**61378**	**61378**

Figure 10.15 : Data format inside the FPGA

The data format shown in Figure 10.15 represents how the data will be transformed inside the FPGA. In order to disable the IIR filter, the output from the decimation filter is sent to the PC. Therefore the higher sixteen bits of the twenty-four bits output is used. To enable the IIR filter, all 24 bit of the output from the decimation filter is sent to the low pass filter.

The transformed data for each channel is available for the following signal processing. Figure 10.16 shows how the synchronization between the FPGA and the current sensor is done. The protocol of the data output from the current sensor is described in Figure 9.8. As mentioned the 'sync' signal will be used for the synchronization and it also contains also the information regarding at which times the data bits are valid. The FPGA is clocked by a 60 MHz signal. The so called 'clk_i' signal triggers the FPGA with every rising edge. This signal is used as the main clock for the whole data process in the FPGA and in the demonstrator tool.

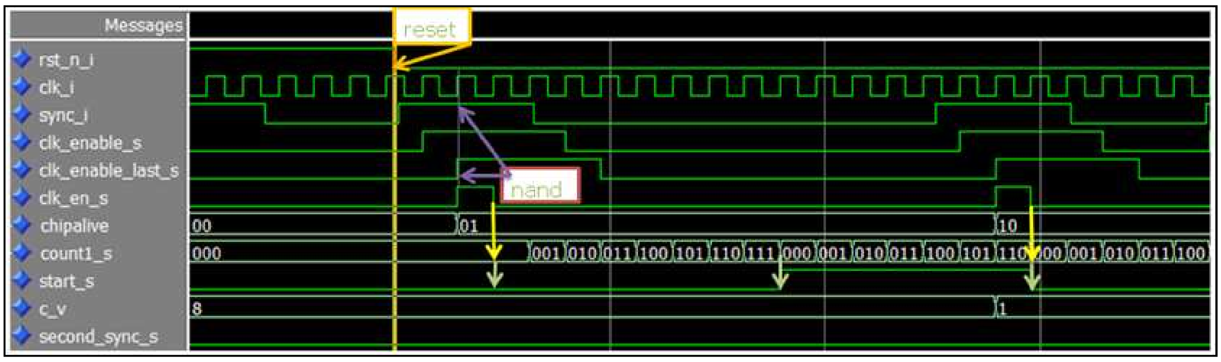


Figure 10.16 : Simulation of the synchronization process

The status of the 'sync_i' signal is transferred to the 'clk_enable_s' signal by every rising edge of the main clock. The status of this signal represents the previous status of the 'sync_i' signal by a delay of one clock cycle. The signal 'clk_enable_last' represents the value of the 'clk_enable_s' signal also with a delay of one 60MHz clock cycle. The signal condition after the 'nand' operation with the signals 'clk_enable_s' and 'clk_enable_last_s' after the next system clock is represented by the signal 'clk_en_s' in Figure 10.16 . This signal is used as the control signal for further data capture. If the signal 'clk_en_s' goes high the first time the 'chipalive' signal will be set to the binary value 01. This will indicate that the FPGA was able to detect the sync signal from the chip and that the synchronization between chip and FPGA has been successful. At this time the red LED on the demonstrator tool begins to flash.   If the 'clk_en_s' signal is high, the signal 'count1_s' will be set to zero with the next rising edge of the system clock. If the ‘chipalive’-signal has changed the status from zero to high, the signal ‘count1_s’ will be increased with every rising edge of the system clock. This signal will be used to find the synchronization gap in the ‘sync’ signal. If the ‘count1_s’ signal reaches a value of seven, a gap in the synchronization signal has been detected. At this time the ‘start_s’ signal changes from the status low to high and indicates the start sequence to capture the data from the data signals with every rising edge from the ‘clk_en_s’ signal. The ‘count_s’ register will be set to zero and begins to count again with the next rising edge of the system clock.

One Period of the sync signal has a duration of 125ns. Therefore the zero time lasts 62.5 ns. The ‘count1_s’ signal will be set to zero at every rising edge of the system clock if the ‘clk_en_s’ signal is high. In the case that a synchronization gap in the ‘clk_en_s’ signal appears, the sync signal will be zero for 3 zero time cycles. During this period the ‘count1_s’ signal reaches the value 7 and sets the ‘start_s’ signal again into the high state. By counting the system clock during the period where the ‘sync’ signal is zero the low time of the synchronization signal, respectively the gap can be estimated. Equation 10.3 describes how the time will be rated by counting the edges of the system clock.

$$zero\_time = \frac{1}{f_{main\_clk}} \cdot clk\_i_{(ri\sin g\_edge)}\Big|_{clk\_en\_s=0}$$

Equation 10.3

$$zero\_time = \frac{1}{60MHz} \cdot 7 = 116.6ns$$

Equation 10.4

$$Periode\_sync\_s = \frac{1}{8MHz} = 125ns$$

Equation 10.5

Figure 10.17 shows how the input values are transferred to the output of the ‘top_current’-instance. After the ‘start_s’ signal becomes high, the register ‘data_left_s’, ‘data_middle_s’ and ‘data_right_s’ receive the values from the corresponding input signals by the rising edge of the ‘clk_en_s’ signal. At each sampling point the variable ‘c_v’ counts the received input bits. When the variable reaches the value seven, the signal ‘second_sync_s’ will be set to high to indicate that one input sequence has been transferred successfully to the FPGA. If the ‘second_sync_s’ signal is high and the next gap in the sync signal occurs, the seven bit register which includes the status of the last input signal will be sent to the output. Also the ‘block_enable_s’ signal will be set to high for one system clock to indicate that new data is available at the output of the top_current instance.

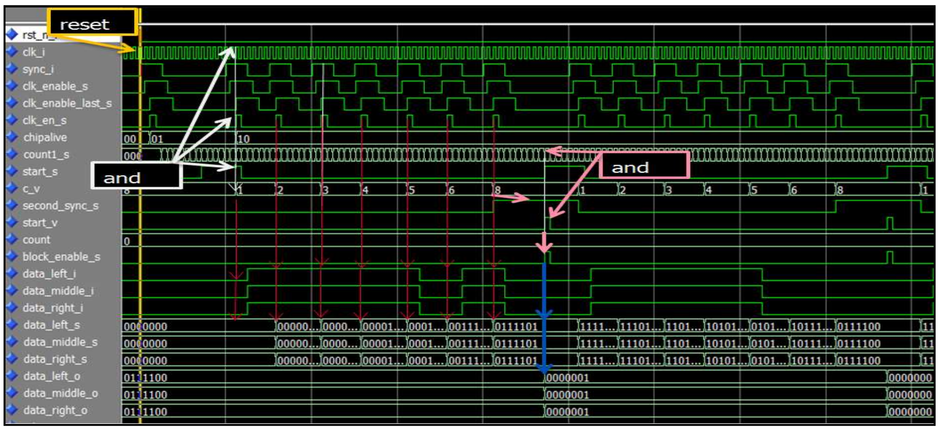


Figure 10.17 : Simulation of rating data from the CLMCS

The data format from the five bit Hall value has a possible range of 0 to 31 in decimal. The values zero and 31 are not valid by design. The zero point value of the sensor is represented as fifteen decimal. Figure 10.18 shows how the output from the CLMCS changes if the sensor detects a magnetic field.

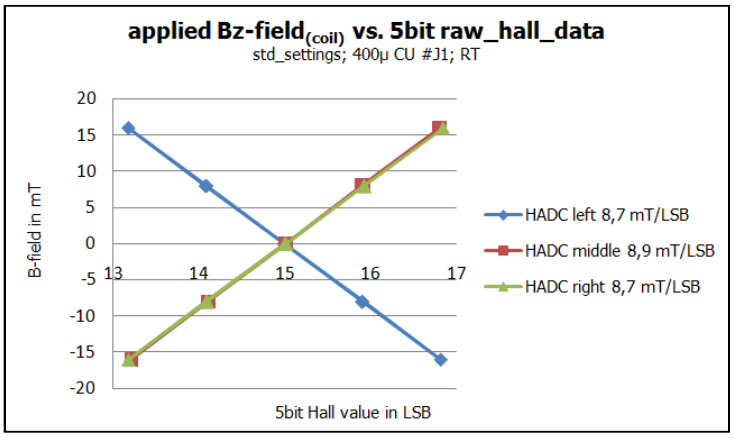


Figure 10.18 : Hall output from the CLMCS depending on applied B-field

10.2.12 Backend interface

The backend interface translates the commands from the user interface and modulates the signal which controls the transistors on the modulation hardware. First of all the command “111000101011” in binary has to be sent to the chip in a defined time window to enable the test mode of the sensor. Figure 9.21 shows the modulated signal to enter the test mode. The command which has to be modulated consists of 2 eight bit signals from the user interface. How the commands are merged together is described in the user interface chapter. Every command but three are able to be modulated. The command which consists of 12 zeros is a nonsense command for the chip and will not be modulated. Also the command which only consists of ones will not be modulated. And the command ‘111100001111’ which is also a nonsense command for the chip will not be modulated. However the command is used to trigger the software reset on the FPGA. If the received data from the chip are not authentic or the communication with the FPGA has been stopped, it would be necessary to reset the hardware, via the software, to automatically start working from its initialized state. Especially when doing the chip calibration by automated measuring, it is important to guarantee the capability to initials a new startup of the measuring should an error occurs. Each command from the backend interface will be modulated twice to the chip. In order to send a read request, to get out some register values from the chip, the clock to read the response is generated by sending the same command again. If the chip is set into the reading mode, the middle channel will be used to read the data from the CLMCS with every generated clock from the backend interface. Sending one command to the chip lasts 2,4ms as shown in the Figure 10.19.

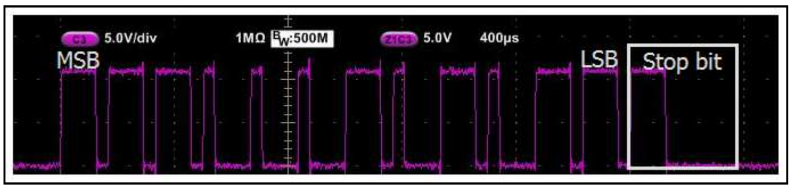
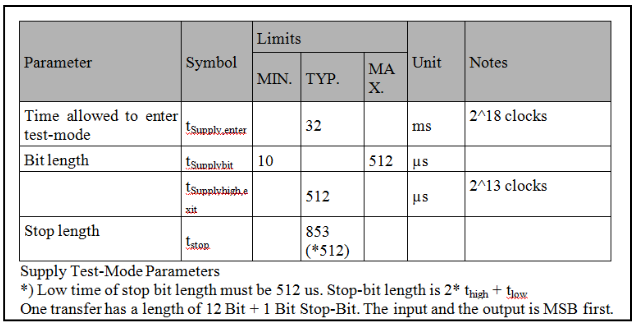


Figure 10.19 : Testmode sequenz modulated on the chip supply

Figure 10.19 shows the command to enable the test mode in the simulation. It is necessary to guarantee that the test mode command appears in a defined time window. Table 10.11 shows the time valid time settings.



Parameter	Symbol	Limits			Unit	Notes
		MIN.	TYP.	MAX.		
Time allowed to enter test-mode	$t_{Supply,enter}$		32		ms	2^18 clocks
Bit length	$t_{Supplybit}$	10		512	µs	
	$t_{Supplyhigh,exit}$		512		µs	2^13 clocks
Stop length	t_{stop}		853 (*512)			

Supply Test-Mode Parameters
) Low time of stop bit length must be 512 us. Stop-bit length is 2 t_{high} + t_{low}
One transfer has a length of 12 Bit + 1 Bit Stop-Bit. The input and the output is MSB first.

Table 10.11 : Time protocol to enter test mode [11]

The only way to check if the chip has entered the test mode successfully is to send the command '111000000001'. This command will set the chip in the read out mode if the test mode has been enabled successfully. If the chip interprets the command correctly, the left and right data channel will be set to zero. By sending the command '111000000000' the read out mode will be disabled and the CLMCS will deliver data on all data lines again. Figure 10.20 shows how the chip is set to reading mode.

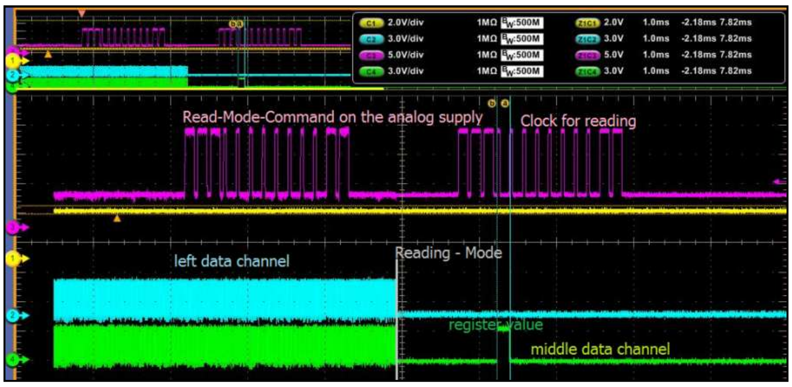


Figure 10.20 : Enabling Readout Mode

Figure 10.21 shows how to disable the reading mode.

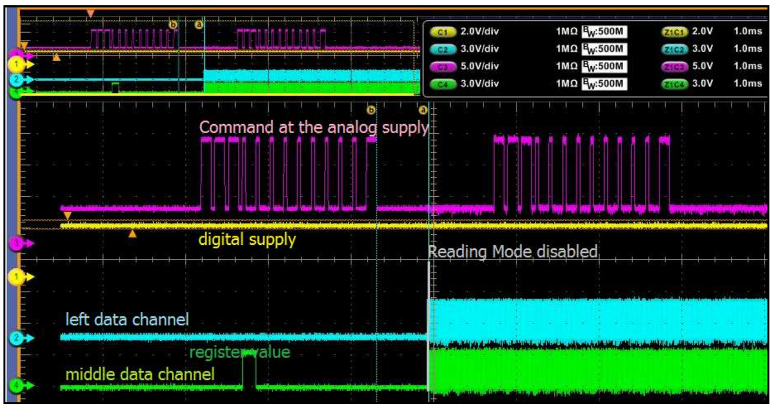


Figure 10.21 : Disabling the Readout Mode

The backend interface is also responsible for a defined start up sequence from the CLMCS. It is important that if a reset occurs, the supply from the digital part of the chip and also the supply of the analog part have to be set under the supply limits for a defined time. Empirical tests give the time period for a successful reset of 80µs. Figure 10.22 shows the supply level of the CLMCS if a reset occurs.

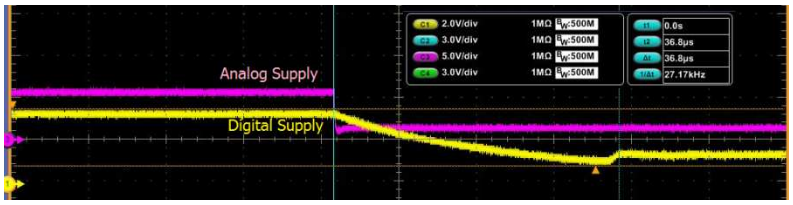


Figure 10.22 : Reset supply level

Figure 10.23 shows the startup sequence after the software reset.

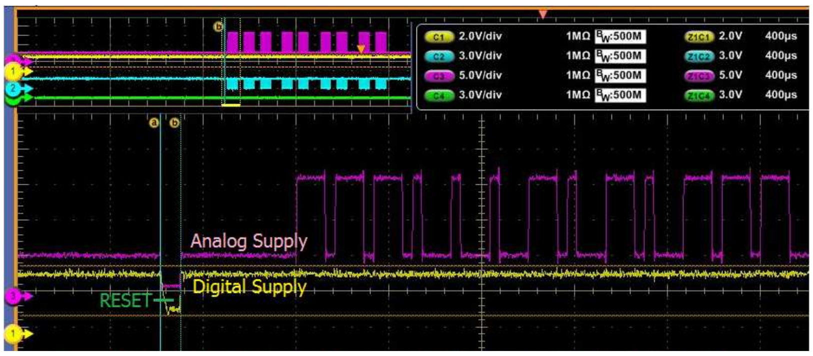


Figure 10.23 : Startup Sequence after Software Reset

The setup commands for the Backend interface of the CLMCS are shown in Table 10.12.

MSB												**Settings**	**Reset-Value**
0	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_{10}-D_0 = Overdrive DAC	"01000000000"
1	**0**	**0**	X	X	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7-D_0 = Hall-Bias (Fuse)	Fused Value
1	**0**	**1**	X	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_1-D_0 = TrimStress D_3-D_2 = TrimHall_TC D_4 = Rvertical (Fuse) D_5 = BE_EXIT_OFF D_6=ODR-Disable D_7 = range_x4	"01" "01" '0' '0' '0' '0'
1	**1**	**0**	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_0 = chop_integ2 D_2-D_1 = Chop static D_4-D_3 = Chop speed D_6-D_5 = test_mux D_7 = st-adc constant D_8 = st-adc source	'0' "00" "00" "00" '0' '0'
1	**1**	**1**	**0**	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_0 = Readout mode D_1 = FBoff D_2 = FB-Freeze D_3 = FB_Decimation D_5 -D_4 = FB_AMP D_6 = FB_hyst_en D_7 = FB_phase_invert	'0' '0' '0' '0' "00" '0' '0'
1	**1**	**1**	**1**	D_7	D_6	D_5	X	X	X	X	X	Readout commad D_7-D_5 = address "000" = overdrive DAC "001" = feedback left "010" = feedback middle "011" = feedback right "1XX" = Fuses (Rvertical&Hall-bias)	

Table 10.12 : Commands for the Backend Interface of the CLMCS [11]

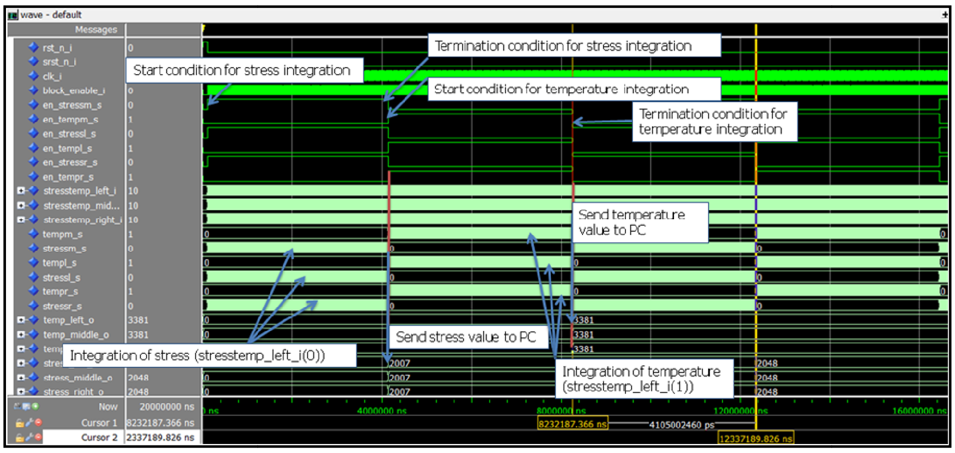
10.2.13 Instance Dut_stress_temp

This entity integrates the one bit stress and one bit temperature signals to twelve bit signals. The chip consists of three identical parts and each of them has a temperature and stress sensor. Each pair of the temperature and stress sensors share one ADC to convert the signal into the digital domain.

Hence it is not possible that both of the two bits have a status of one at the same time. This behavior allows integrating the correct number of temperature or stress bits. Figure 10.24 shows the termination condition which is also the start requirement to integrate the complimentary signal. At the very beginning the FPGA senses the two lost significant bits of the three input signals from the CLMCS. If one of them changes the status from zero to one, the FPGA begins to integrate the bits of the position where the status change has occurred. Each bit at this position will be integrated every 1µs until a status change at the other bit position is detected. If the bit at the temperature position changes its status from zero to one, all following bits will be integrated until for example a bit at the stress position will change its status. At this time the FPGA changes from the 'temperature-integration' mode into the 'stress integration - mode'. That means that the integrated value, which is stored in a twelve bit temperature register will be sent to the PC and then set to zero. At the same time the integration of the stress bits starts. The value will be integrated and stored in the stress register until a bit at the temperature position changes to one again. After a detection of a one at the temperature position during the integration of the stress values, the twelve bit stress register will be sent to the PC and then set to zero. This process happens for each of the three channels simultaneously. Due to the multiplexing of the temperature and stress value, the data update rate u_R occurs approximately every 2^{12} micro seconds.

$$u_R = 2^n \cdot \frac{1}{fs} = 2^{12} \cdot \frac{1}{1000000} \approx 4.1ms$$

Equation 10.6



wave - default
Termination condition for stress integration
Start condition for stress integration
Start condition for temperature integration
Termination condition for temperature integration
Send temperature value to PC
Send stress value to PC
Integration of stress (stresstemp_left_i(0))
Integration of temperature (stresstemp_left_i(1))
Cursor 1 8232187.366 ns
Cursor 2 12337189.826 ns

Figure 10.24 : Simulation of reading temperature and stress from the CLMCS

Figure 10.24 shows the simulation of the 'stresstemp' instance. Initially the bit at position zero changes from the status zero to one at the input signals 'stresstemp_middle_i', 'stresstemp_right_i' and 'stresstemp_left_i'. This is the start condition to integrate all bits at the position zero until one bit at position one changes its status from zero to one. At the time position 4ms, the input from the stress register will be sent to the interface. At the same time the register will be set to zero. Due to the status change of the temperature bit at the time position at 4ms, the FPGA begins to integrate the temperature bits until a bit on the stress position at the input signal changes its status at the time position 8ms. At this point in time the integrated temperature value will be sent to the output and the temperature register will be set to zero again.

10.2.14 Decimation Filter

The signal from the Hall plate will be converted to a five bit data stream by a continuous sigma delta ADC. This five bit data stream has to be converted into a signal with gives a higher accuracy. Therefore a decimation filter is used to increase the bit length of the Hall signals. Because of decimation, the sampling rate of the data stream will be decreased by the down sampling factor.

The decimation filter receives the input data directly from the 'currrent_top' instance. These signals have a bit length of five. The input data has a signed format. The internal register length is dependent on the down sampling rate as well as the register length from the input signal as it is described by Equation 8.6. Therefore the register length of the intern register has been set to 29. The used Hogenauer-filter is explained in detail in the filter chapter. Figure 10.25 shows the structure of the filter.

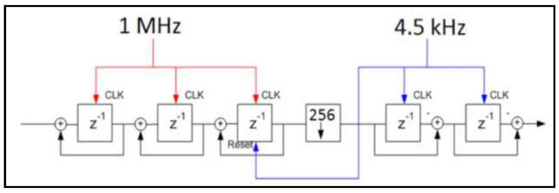
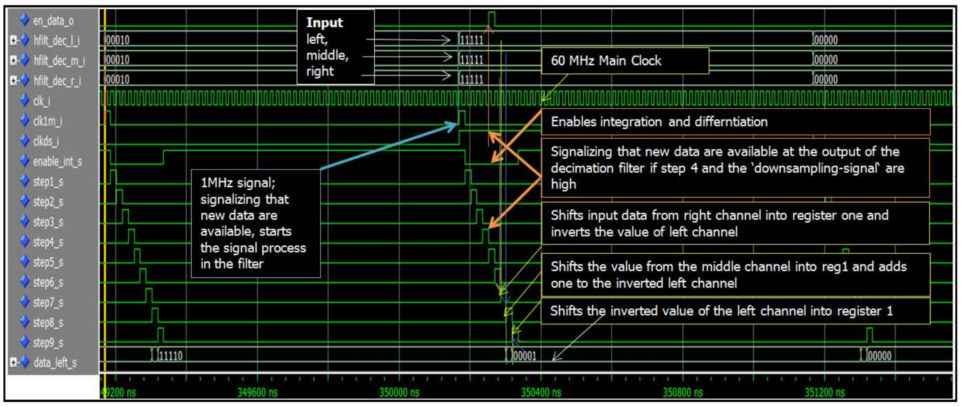


Figure 10.25 : Structure of the implemented CIC - filter

With the intention of adapting the filter design for the final product, it is important to design structures which do not waste space on the chip. To save space and to ensure an efficient signal process, shared adders for all three channels have been used. Figure 10.26 describes the signal flow inside the FPGA.



1MHz signal; signalizing that new data are available, starts the signal process in the filter

Enables integration and differntiation

Signalizing that new data are available at the output of the decimation filter if step 4 and the 'downsampling-signal' are high

Shifts input data from right channel into register one and inverts the value of left channel

Shifts the value from the middle channel into reg1 and adds one to the inverted left channel

Shifts the inverted value of the left channel into register 1

Figure 10.26 : Data process at the decimation filter

The data process inside the filter is triggered by the 1MHz signal which depends on the sampling rate preset at the current sensor. Setting the main operation clock at 60 MHz and with a data sampling rate of 1MHz, provides enough time to allow for shared adders for each of the three input channels. The data handling is split into three main steps which each consist of three sub steps. Therefore each register is split into three sub registers as shown in Figure 10.27.

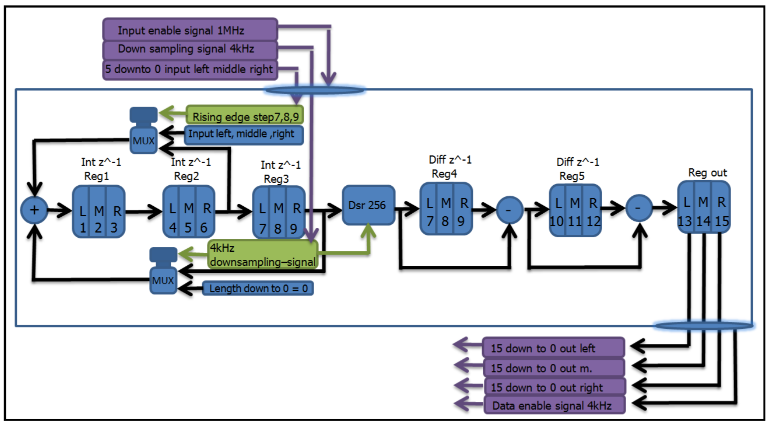


Figure 10.27 : Structure of the implemented decimation filter in the FPGA

After three clock cycles, the register is fully loaded with all values from the three different channels. At the initialization, all six registers are set to zero, triggered by the reset signal. New data are available with each 1MHz signal. In the first 6 steps the values of the 3 integrator registers will be loaded with the actual register values. In these steps the multiplexer connects the output from the second register with the adder at the first register. In the next three steps the multiplexer in Figure 10.27 connects through the Hall input signals and loads the register with the actual Hall value. The filters work with a down sampling rate of 256. That means the signal ‘clkds_s’, which triggers the down sampling process, occurs with a frequency of 4 kHz.

Every time the down sampling signal is synchronous with the rising edge of the step 4 signal in Figure 10.26, a down sampling process will be triggered. In this case the contents of register three will be reset and consequently initialized with zero, that means that the multiplexer in Figure 10.27 connects the zeros register to the adder instead of the contents of register three. At the same time register four at the differentiator segment will be loaded with the previous value of register three from the integration segment.

To load register five, the previous value from register four must be deducted from the current value in register four. At this trigger point the difference of the current value in register five and the previous value from register five is available at the output of the decimation filter after the down sampling process.
A control signal indicates that new data from the decimation filter is available to be sent via USB to the PC or stored in the SDRAM before sending them to the user interface. Only the 24 upper bits will be used in the further signal process. The structure of the Hall plates on the test chip is adapted from the Infineon linear Hall sensor TLE4997. The current chip includes three of these Hall sensors. The layout of the left Hall plate is mirrored. That is because the data stream from the left channel has to be inverted to ensure correct data interpretation. The inversion of the left channel happens in the decimation filter at the seventh and the eight steps. As mentioned before the registers will be loaded with new Hall values in the last three steps of the nine steps with data in a signed format. The upper bits in the register will be filled with the most significant bit from the input. The four lower bits will be loaded with the four least significant bits from the input. At step seven the input from the right channel, which is already in a format as described, will be shifted in register one and the five bit input from the left channel will be saved in a help register.
The input from the middle channel will be shifted in register one in step eight. In this step, one bit will be added to the help register which includes the inverted five bit value from the left input signal. In the last step the corrected value of the left channel will be shifted from the help register into register one.

10.2.15 IIR filter

The output data stream from the decimation filter will be filtered by a low pass filter first or second order. The input signals have a length of 24 bit, like the output from the decimation filter which is the pre-state of the low pass filter. The intern register length in the low pass filter goes from 32 down to zero. There is also a five bit input signal to adjust the cut off frequency. The FPGA structure allows cascading two low pass first order filters by sending the corresponding command from the PC to the user interface via USB. The -3dB - frequency is individually adjustable for both. The structure of the used low pass filter is shown in Figure 10.28.

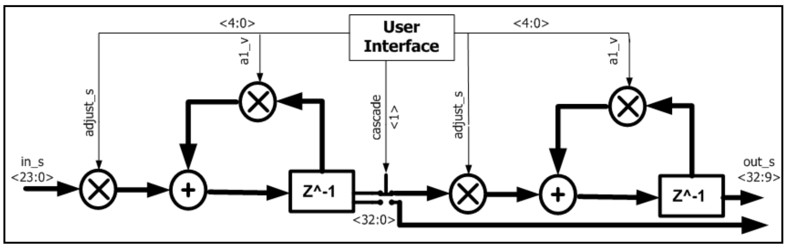


Figure 10.28 : Structure of the implemented low pass filter

The figure shows an IIR filter first order. Dependent on the users command the output from the first filter will be connected to the input of the second IIR filter. The values of the multipliers depend on the adjustable signal 'adjust_s'. The length of this signal goes from four down to zero. In theory it is possible to set up 32 cut off frequencies for each filter. The multiplier variable a1_v represents the reciprocal value of the 'adjust_s'– signal. The accuracy depends on the register length that is used for the calculation inside the low pass filter.

Figure 10.29 shows how the signal process is done.

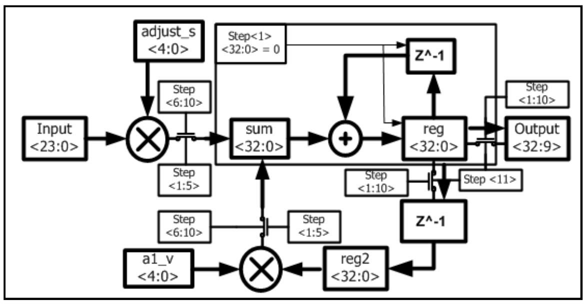


Figure 10.29 : IIR signal process

The input register length of the low pass filter will be calculated as follows. Firstly the 24bit input signal has to be multiplied with the settable multiplication coefficient. The 'adjust_s' signal in Figure 10.28 and Figure 10.29 represents the multiplication coefficient and has a length of five bit. The multiplication in the FPGA is separated into five steps. Dependent on the value of the 'adjust_s' signal, the input value has to be shifted to the left, which relates to a multiplication with a multiple of two. Therefore the lower bits have to be filled with zeros and the upper ones should always have the same sign bits. Because of the length of the 'adjust_s' signal the maximum number of shift operations due to a multiplication can be four. This is why the length of the result register has to be enlarged by five. The input value will be enlarged with five sign bits. The 24 bit input will be transferred into a help register with the number of 29 in a signed format. The first five bits are representing the sign of the 24 bit input value. The rest of the help register is filled with the input value. This signed value with the length of 29 will be saved in the register 'reg2' after adding the previous values of this register. The input of the register 'reg2' has to be initialized with zeros. In the second step of the filter design, the previous value which represents the result from the multiplication of the input value and the 'adjust_s' signal has to be multiplied with the 'a1_v' coefficient. This coefficient also consists of five bits and represents the inverted value of the 'adjust_s' signal. The value from the register 'reg2' also has to be enlarged by the number of five to ensure no loss of information if the value of the register will be multiplied by the 'a1_v' register. It is not possible to change the register size online because the code is done in hardware and cannot be changed after the synthesis. To avoid an overflow by the second multiplication, the value from the register reg2 also has to be enlarged with five bits. In the same way as described the multiplication with the previous value from the register reg2 and the a1_v register has to be done. But because of the fixed register length the LSB will be lost by shifting the sign bits from the MSB to the LSB. Thereby a loss of resolution occurs. To improve the granulation, all registers will be enlarged on the lower side before the syntheses is done. The number of zeros should be the same as the number of the used signed bits for the multiplication with the previous result. Therefore the intern sizes of the registers are set to 33 in the used low pass filters.

10.3 Demonstrator

10.3.1 Software installation

The hardware was tested on windows 2000 and windows 7. To install the device, the 'usbser.sys' file should be used. Therefore a custom '.inf'-file is necessary to tell windows that it must use this driver. After windows has installed the driver correctly the device appears as a serial com port in the hardware manger. After the installation the communication with the device can be done as described in the interface chapter. The appendix includes the used setup information file. [23]

10.3.2 Representation

Figure 10.30 shows the representation interface of the demonstrator tool to represent the measured magnetic field, respectively the measured current.

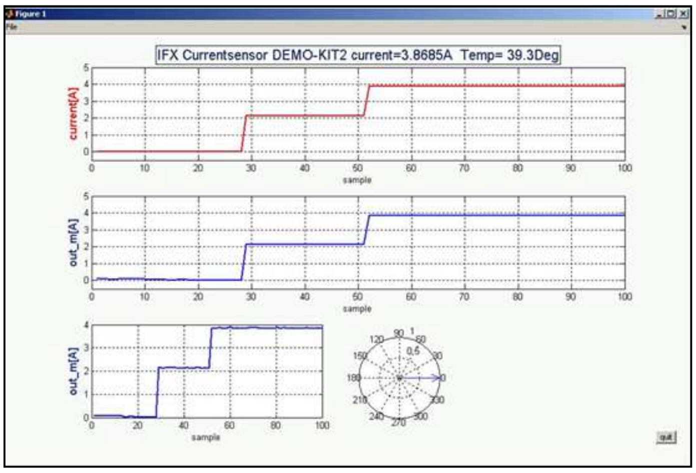


Figure 10.30 : Current representation interface

The calculated current is represented in the top of Figure 10.30. The determined current which has been calculated by only one channel is shown in the middle of the figure. Therefore the middle section may also show the influence of an interfering field because the current calculation is only done with the data from one Hall channel. At the lower side of Figure 10.30 the scope has a higher resolution to represent the high sensitivity of the sensor.

11 Measurements

11.1 Noise measurement and plausibility check of the FPGA data process.

Dependent on the application where the usage of the Triple Hall current sensors will be used, the amount of noise in Ampere, respectively the noise in Tesla will be of note. By calculating the noise of the sensor a performance check of the data process inside the FPGA has also been done. To check the correlation of the results between the 16 bit output, after the signal processing in the FPGA and the raw five bit Hall data, the following measurements have been done. To get information about the noise of the three Hall channels, the five bit Hall data from the CLMCS have been measured with a logic analyzer. Afterwards the measured data have been analyzed with the spectrum analyzer software in MATLAB. This software allows to analysis spectra by transforming the raw data into the frequency domain by using the Fast Fourier Transformation. The software was developed by Infineon and can be used for data analyzes. Figure 11.1 shows the spectrum of the three channels after the Fourier Transformation of the five bit Hall data measured by the logic analyzer.

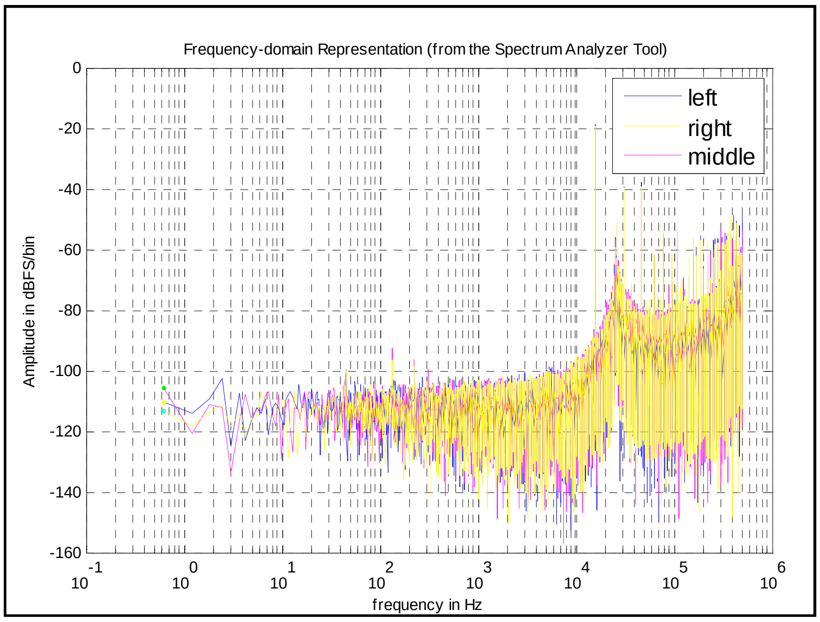


Figure 11.1 : Spectrum of the five bit Hall data from the CLMCS

Figure 11.2 shows the spectrum of the right channel.

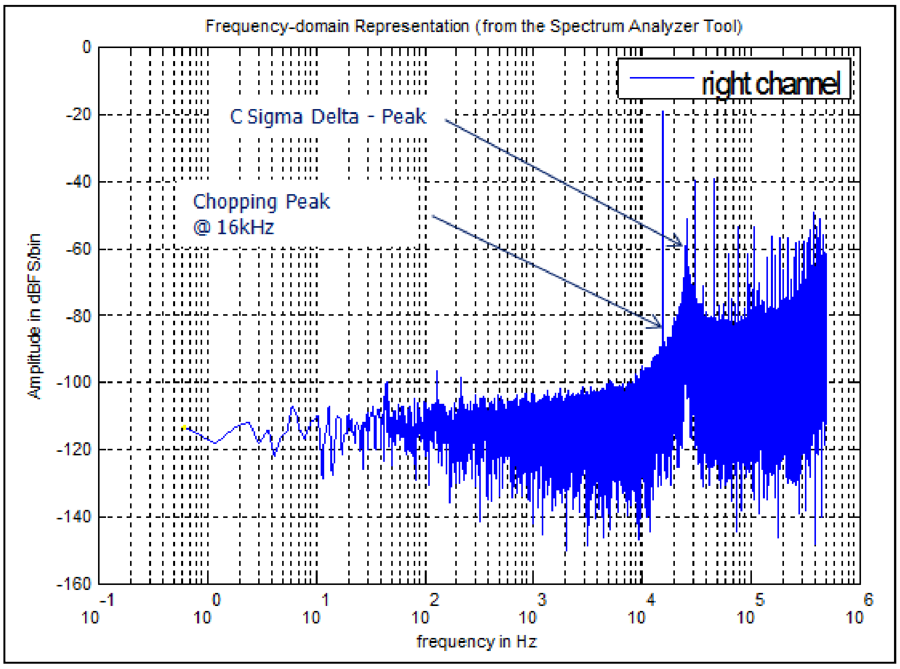


Figure 11.2 : Spectrum of the five bit raw data from the right channel

The noise floor for the particular bandwidth has been calculated with the spectrum analyzer software in dB. Table 11.1 includes the calculated noise in dB at the specified bandwidths.

Band width Hz	**Noise dBFS**	**Noise dBFS**	**Noise dBFS**
10	-99,8	-100,5	-102,1
14	-97,7	-99	-99,7
20	-95,6	-96,2	-97,9
50	-91,2	-91,6	-92,5
500	-81,3	-81,4	-82
1000	-78,5	-78,5	-79
2000	-75,7	-75,5	-75,9
5000	-71,4	-71,2	-71,4
50000	-30,6	-29,2	-31,2
200000	-29,1	-28	-29,5
500000	-18,4	-18,9	-18,7

Table 11.1 : Calculated noise in dBFS depending on the bandwidth

To check the correct operating mode of the FPGA data process, the five bit Hall data from the CLMCS has been buffered in the SDRAM on the micro module before they have been sent to the PC via the USB interface. Afterwards the data has been analyzed with the spectrum analyzer software in the same way as the data which has been measured by the spectrum analyzer. Both data sets came to the same spectrum, respectively noise floor, which verifies the correct data flow from the FPGA to the PC.

In the next step the Hall sensitivity of the same CLMCS device has been determined. Therefore a magnetic field gradient has been applied to the chip and the Hall data has been read and processed by the FPGA before sending them to the PC via the USB interface. Figure 10.18 shows the applied magnetic field in mT versus the measured five bit HADC values of the three channels. In the next step the noise in µT has been calculated as follows. Equation 11.1 calculates the full-scale range in mT from each channel of the sensor.

$$FS_{mT} = 2^{n_{bit}-1} \cdot \frac{15}{16} \cdot \frac{dB}{dHALL_{LSB}}$$

Equation 11.1

Table 11.2 shows the calculated full-scale range in mT.

FS in mT		
left	middle	Right
-131,542	133,604	130,7937

Table 11.2 : Full-scale in mT for each channel

Equation 11.2 shows how the noise has been calculated depending on the data from the spectrum analyzer and the sensitivity measurements of the sensor.

$$Noise_{\mu T} = \frac{1}{10^{\frac{Noise_{dB}}{20}}} \cdot \frac{FS_{mT} \cdot 1000}{\sqrt{2}}$$

Equation 11.2

Figure 11.3 shows the calculated noise in µT depending on the respective frequency.

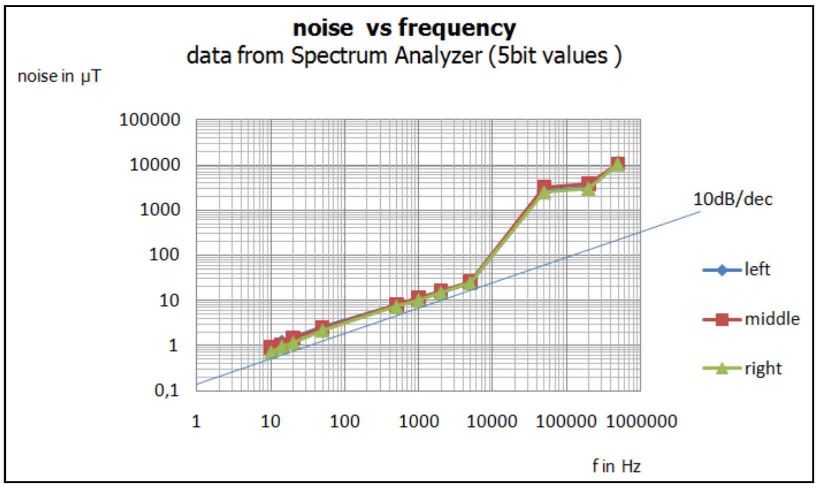


Figure 11.3 : Noise in µT depending on the frequency

The increase of the 10dB/Dec-line at 10 kHz depends on the chopping peak which can be seen in Figure 11.2 at 16 kHz. This peak at 16 kHz depends on the chopping frequency of the spinning current section of the CLMCS. The peak at 22 kHz in Figure 11.2 depends on the sigma delta modulator of the CLMCS sensor.

To check the signal process of the FPGA the noise of the sensor has been calculated in a different way, to compare the results independent of each other. Therefore the Hall sensitivity of the CLMCS has been determined by reading the 16 bit HADC values after the decimation and low pass filter in the FPGA. Figure 11.4 shows the Hall sensitivity depending on the applied magnetic field which has been produced by the Helmholtz coil.

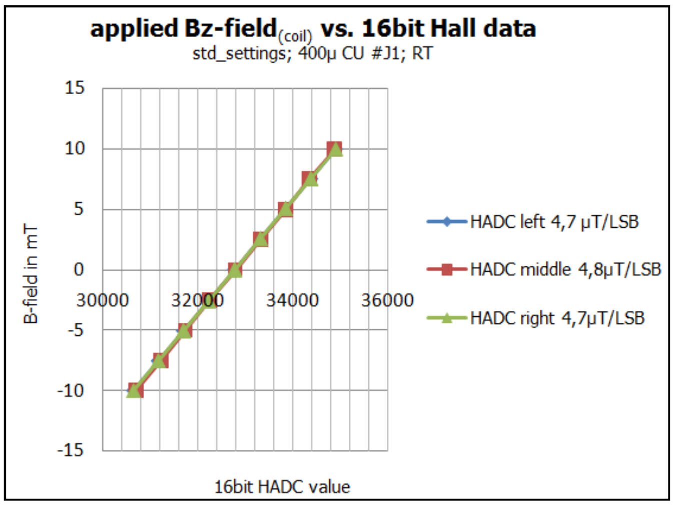


Figure 11.4 : Hall sensitivity depending on the applied B – field produced by the HH-Coil

As described in the software chapter, there is an IIR low pass filter connected in series to the comb filter. The cutoff frequency of this low pass filter has been set via the user interface to the corresponding frequency values, which were used for the noise floor calculation with the spectrum analyzer. The standard deviation of the 16 bit values has been calculated for each channel as described in Equation 11.4.

For each cutoff frequency setting 4000 sampled Hall values of each channel from the FPGA has been read for the calculation.

$N = 4000$ Number of sampled values for each measurement

$$\overline{x}_{left} = \frac{1}{N}\sum_{i=1}^{N} x_{i_{left}}$$

Equation 11.3 [3]

$$\sigma_{left} = \sqrt{\frac{\sum \left(x_{left_i} - \overline{x}_{left}\right)^2}{N-1}}$$

Equation 11.4 [3]

The calculations for each channel have to be done separately.

Table 11.3 includes the sigma of the channels corresponding to the adjusted cutoff frequency of the measured HADC data.

_-3dbf in Hz	sigma in lsb		
f_cut_off	left	middle	Right
22	0,382904	0,526823	0,490885
46	0,568044	0,723502	0,562878
70	0,587218	0,849565	0,636637
94	0,685475	0,931458	0,742656
120	0,776349	1,067145	0,853811
396	1,165893	1,644629	1,315459
433	1,182436	1,723523	1,309219
512	1,775303	2,565588	2,058917
4445	1,824761	2,605046	2,093118

Table 11.3 : Sigma of the Hall data depending on the IIR setting

Table 11.4 shows the calculated noise in µT by using the Equation 11.6.

$$FS_A = 2^{n_{bit}-1}\frac{15}{16}\cdot\frac{dI}{dHALL_{LSB}}$$

Equation 11.5

$$Noise_{A_{left}} = \frac{\sigma_{left}}{2^{n_{bit}-1}\cdot\frac{15}{16}}\cdot\frac{FS_{A_{left}}}{}$$

Equation 11.6

_-3dbf in Hz	noise in µT		
f_cut_off	left	middle	right
22	1,768131	2,368034	2,232791
46	2,623052	3,25209	2,560254
70	2,711592	3,818735	2,895745
94	3,165309	4,18684	3,377973
120	3,584937	4,796741	3,883561
396	5,38373	7,392491	5,983368
433	5,46012	7,747116	5,954984
512	8,197798	11,53214	9,364989
4445	8,426178	11,7095	9,520552

Table 11.4 : Noise of the Hall signal in µT depending on the IIR setting

Also the Hall sensitivity has been measured by applying a magnetic field gradient and reading the 16bit HADC values from the FPGA. The sensitivity dependent on the magnetic field which has been produced by a load current within the range from zero to ten amperes has also been measured.

Figure 11.5 shows the Hall sensitivity depending on the applied current.

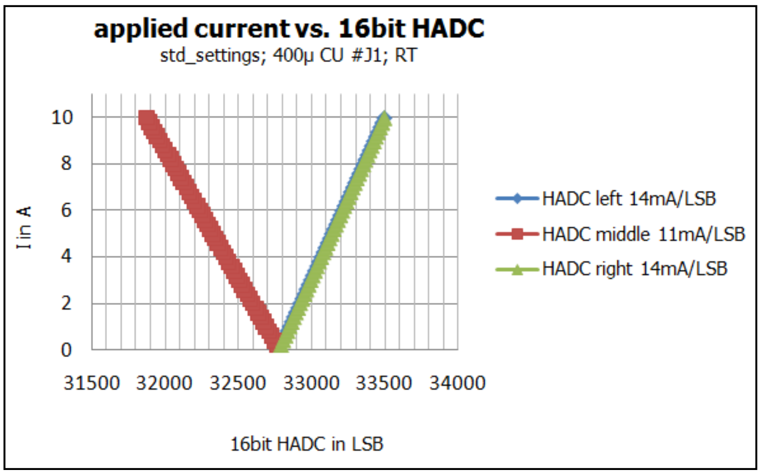


Figure 11.5 : Sensitivity in mA pro 16bit Hall LSB

Table 11.5 shows the calculated full-scale range in A and mT.

FS in A			
left	middle	right	m+(l+r)/2
429,5301	-337,96	427,574	-188,951
FS in mT 16bit			
left	middle	right	m+(l+r)/2
145,03	147,9108	145,0892	73,23566

Table 11.5 : Full-scale range in amps and mT

The first plausibility check can be done by comparing the value of the full-scale range in mT with the two different measurements. There is only a marginal error between the two results which suggest a correct data process in the FPGA. In the next step, the noise in µT, dependent on the calculated FS and the standard deviation of the measurements from the filtered 16bit Hall data, at the specific cut off frequencies, has been calculated as follows.

Figure 11.6 shows the calculated noise depending on the frequency.

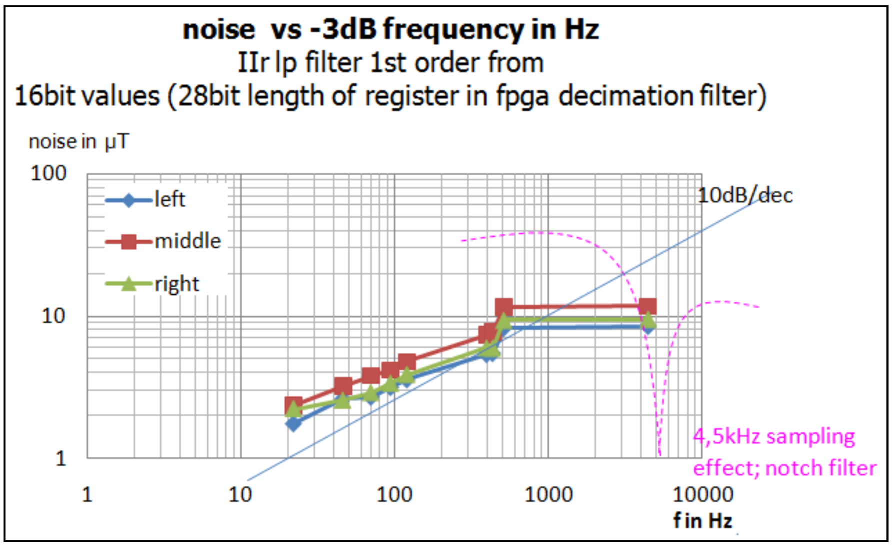


Figure 11.6 : Noise in µT depending on the IIR settings

The decrease of the 10dB/Dec in Figure 11.6 at 4 kHz represents the sampling effect of the used notch filter. This effect can also be seen in Figure 8.4 which represents the spectrum of the used decimation filter.

The accurate correlation of the results is also verified by comparing the noise in µT of the two different measurements. Figure 11.7 shows the calculated noise in ampere from the sensor for each channel separately.

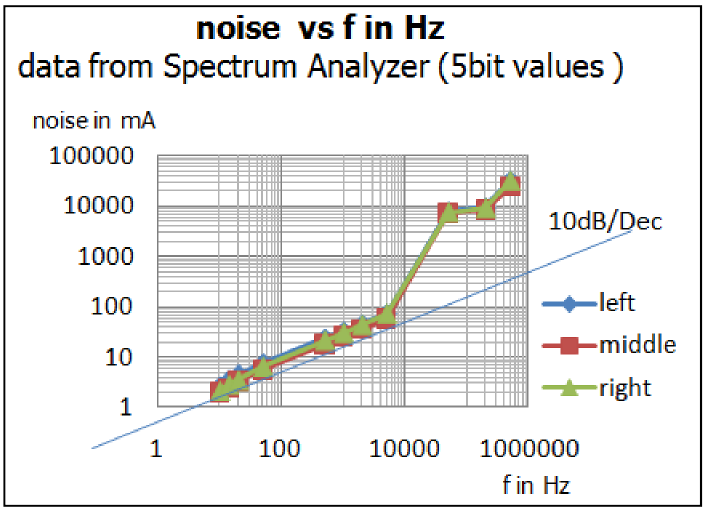


Figure 11.7 : Noise in mA depending on the frequency

Equation 11.7 describes how the complete noise in ampere has been calculated.

$$Noise_A = \sqrt{\left(\frac{Noise_{A_{left}}}{2}\right)^2 + Noise_{A_{middle}}{}^2 + \left(\frac{Noise_{A_{right}}}{2}\right)^2}$$

Equation 11.7

Figure 11.8 describes the noise in amperes of the CLMCS sensor, calculated with the sigma value after receiving the filtered data from the FPGA. The noise in mA is similar to the values from the noise calculation based on the five bit data.

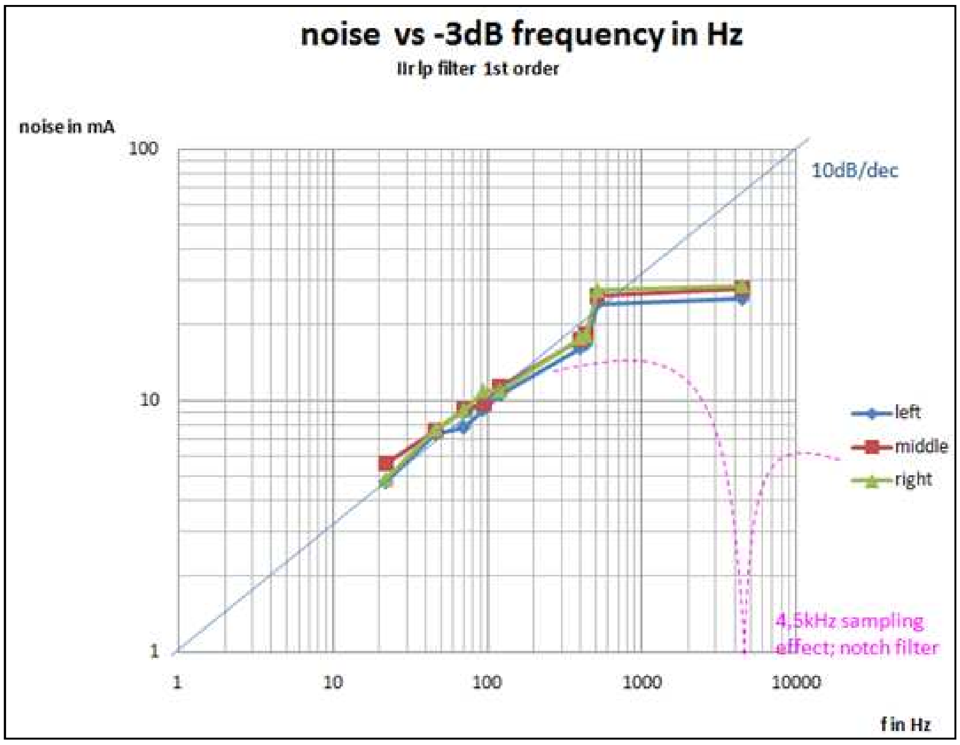


Figure 11.8 : Noise in mA depending on the cutoff adjusted cut off frequency

Table 11.6 shows the application where current sensing is attractive, depending on the required band width range.

Application depending on band width	
5Hz	Energy Management
10Hz	12V Battery Management
1000Hz	Smart Metering
100kHz	Motor and Drivers

Table 11.6 : Application for the current sensor depending on the band width

The spectrum analysis of the sensor has also been done to find the ideal settings which guarantee a small noise level and also a small offset. Figure 11.9 shows the block diagram of the Hall data process from the CLMCS. The feedback loop in Figure 11.9 is used to reduce the offset amplitude from the Hall plate.

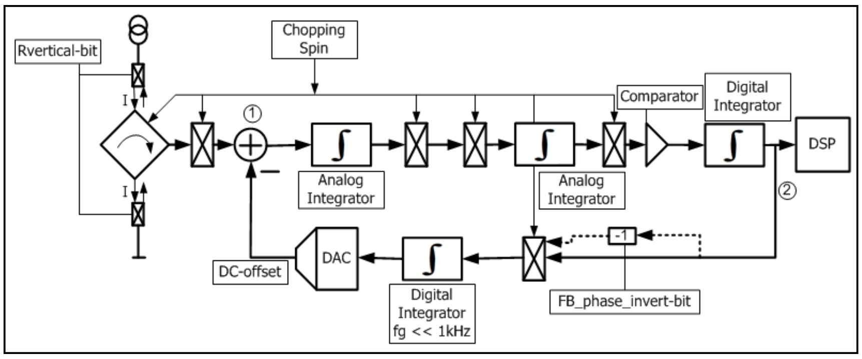


Figure 11.9 : Block diagram of the Hall data process

The spectrum has been measured at the point two in Figure 11.9. If the feedback loop is activated, the offset signal is feed back to the input. At point one in Figure 11.9 the dc-offset will be subtracted from the Hall offset. The integrator in the feedback loop describes a low pass with a cut off frequency in the range of a few Hertz. Due to the low pass in the feedback loop the residual feedback signal at point 1 represents the dc-offset of the Hall plate.

The 16 kHz peak in Figure 11.10 represents the chopping amplitude at point two of the data process chart in Figure 11.9. If the feedback loop has been enabled and the amplifier settings for the feedback loop have the correct value, the offset can be reduced as shown in Figure 11.11. To decrease the offset of the system, two further chopping phases has been implemented. Therefore the supply current direction of the Hall plates will be inverted if the 'Rvertical"-bit from the backend interface will be set. The improvement of the offset by inverting the signal with the 'Rvertical'-bit is described in the next chapter.

Because of the usage of the described chopping mode, a setup has to be determined where the offset amplitude as well as the noise is small and the noise in the frequency domain is independent of the status from the 'Rvertical'-bit. Table 11.7 shows the determined settings and the noise, respectively the attenuation of the chopping amplitude.

HallBias	TrimStress	TrimHall_Tc	Rverital	Chop_integ2	FBoff	FB Freeze	FB Decimation	FB_hyst_en	FB_phase_invert	ODR_Disable	FB_AMP <5>	FB_AMP <4>	Noise [dBFS] BW 1 – 1KHz	ENOB	Attenuation 16kHz Chopping amplitude [dB]
20	0	3	0	0	1	0	0	0	0	0	0	0	-78.8	12.7	ca -20
20	0	3	0	0	0	0	0	0	0	0	0	1	-79.8	12.9	ca -55
20	0	3	1	0	0	0	0	0	0	0	0	1	-77	12.4	Ca -50

Table 11.7 : Feedback loop settings

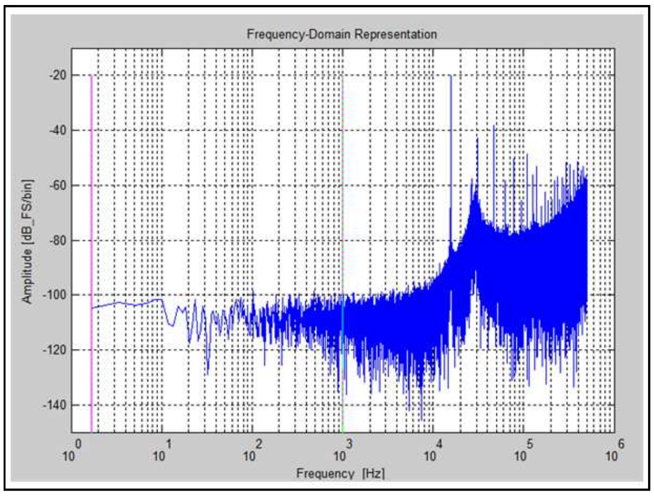


Figure 11.10 : Spectrum with feedback loop off setup

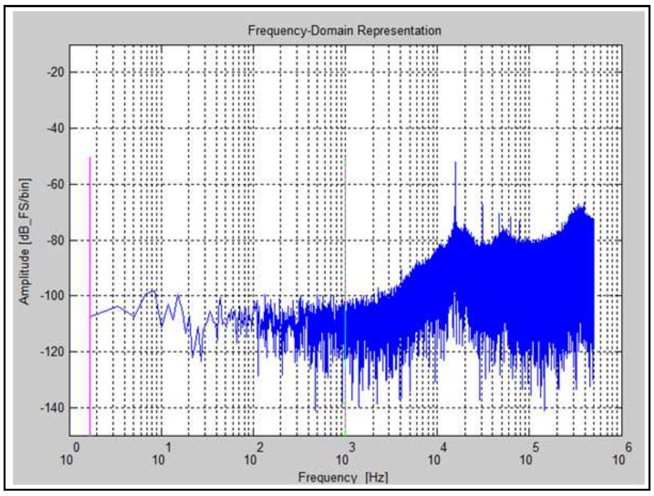


Figure 11.11 : Spectrum with reduced chopping amplitude (feedback loop = on)

The results verify that the feedback loop works correct and that the dc-offset of the Hall plate decreases if the feedback loop is enabled. If the feedback loop is disabled the chip works in a more stable state with the disadvantage of a higher offset in the spectrum. In the redesign of the CLMCS the system coefficients of the described signal path have to be readjusted to ensure a stable operating mode for all channels if the feedback is enabled.

11.2 Bias and operating mode setting

The ideal bias voltage supply for the Hall plates has to be evaluated before the test chip can be used for measurements with high accuracy. Therefore a compromise between a small Hall offset, stability and a flat sensitivity function has to be found. Dependent on the settable register values of the sensor, the Hall bias voltage can be set in the range between 1.4 and 3.0 volts. Also the temperature coefficient of the Hall plate can be set via the backend interface. The adjustable temperature coefficient has a main influence to the behavior of the sensitivity dependent on the temperature. The adjustments for the stress compensation can also be done via the backend interface. Table 10.12 includes all commands for the backend interface. The next chapter describes which commands have been used to find the ideal bias settings.

11.2.1 Hall bias voltage

The Hall plates are supplied by a current which depends on the adjusted bias voltage and the poly resistors in Figure 11.12.

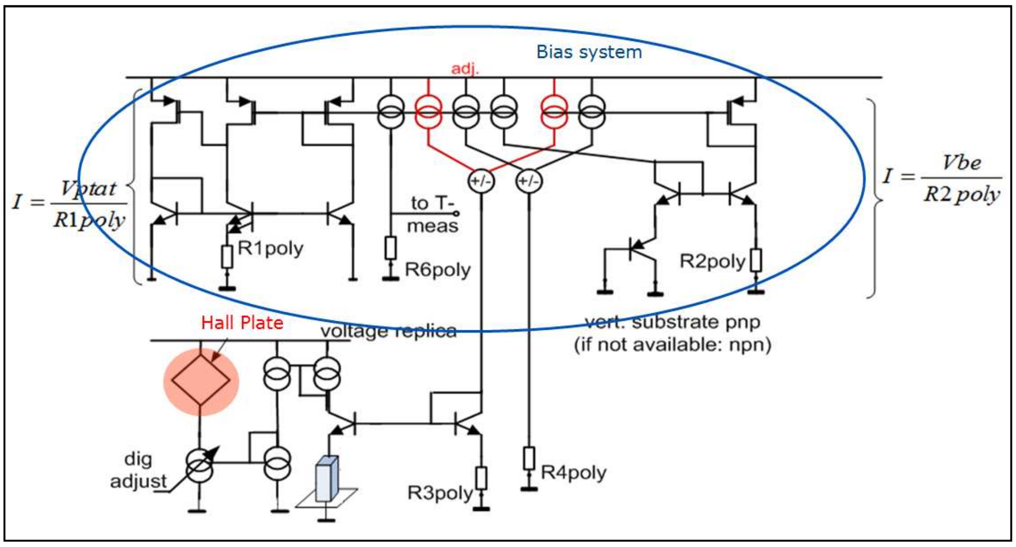


Figure 11.12 : Bias system and stress compensation [9]

Dependent on the bias voltage the sensitivity and the offset of the Hall plates have a different behavior. Figure 11.13 shows how the Hall bias voltage depends on the adjusted bias code.

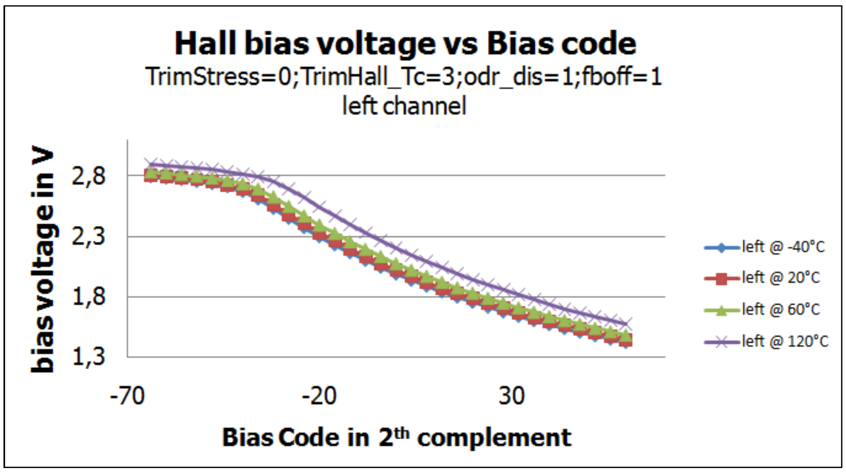


Figure 11.13 : Hall bias voltage versus bias code

The command to set the Bias voltage starts with the binary prefix ‘1000’. The lower byte of the command is used to setup the bias voltage. Figure 11.14 shows the sensitivity in LSB/mT of the Hall plate dependent on the temperature.

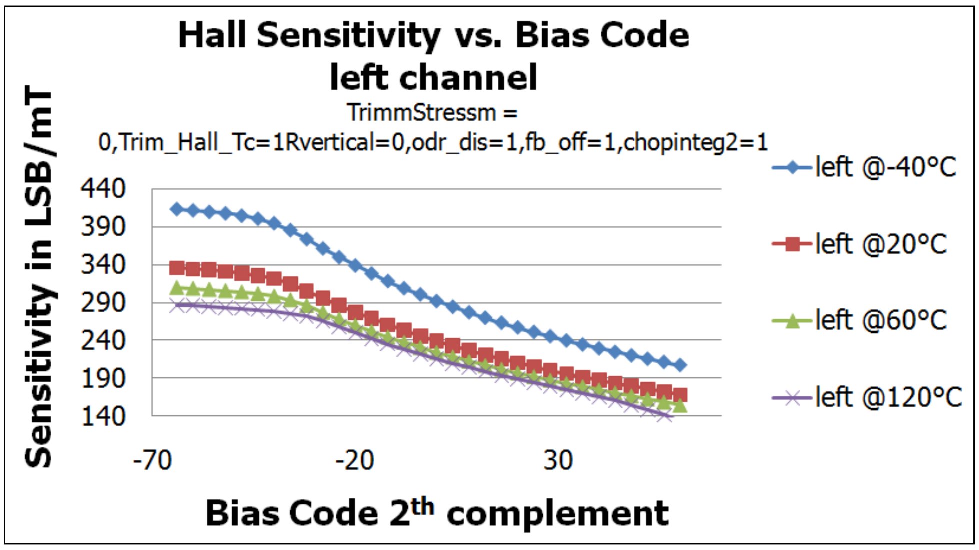


Figure 11.14 : Hall sensitivity versus bias code dependent on temperature

Figure 11.13 and Figure 11.14 indicate that the sensitivity is direct proportional to the adjusted bias voltage. That means, if the bias voltage has a higher level, the sensitivity of the Hall plate increases and therefore the noise decreases as long as the bias voltage stays under the saturation level. Figure 11.15 represents the behavior of the offset versus the Hall bias code.

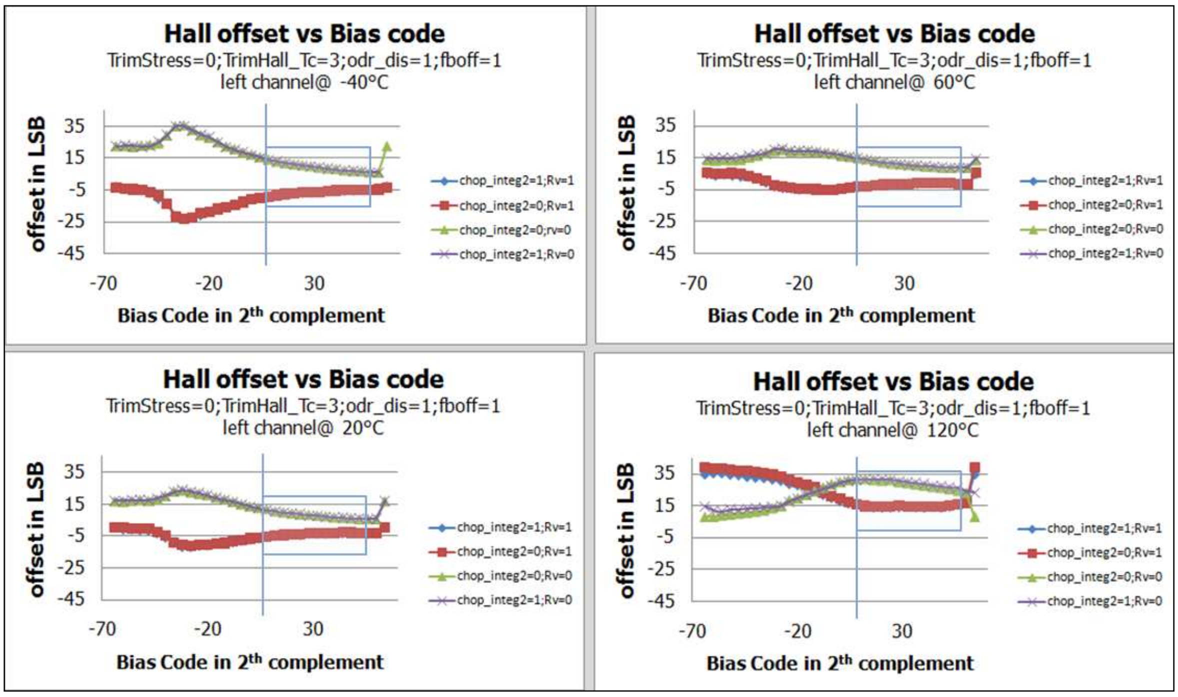


Figure 11.15: Hall offset versus bias code depending on the adjusted temperature coefficent

Figure 11.15 indicates that the system will become more unstable for the bias code in a range from 64 to 0. In this range the voltage is close to the full range as shown in Figure 11.13. The marked areas in the charts from Figure 11.15 are indicating the stable range of the chip dependent on the bias code. Because of the chopping mode from the HADC the output of the ADC is inverted by setting the 'Rvertical' bit. This is shown in Figure 11.15. This figure represents the output of the Hall sensor, respectively the output of the digital chopping demodulation. In the redesign of the current sensor, the 'Rverical'-bit from the command interface is used to demodulate the signal from the chopping stage of the ADC. Thereby the signal is inverted at the output. This can be seen in figure Figure 11.15. Each time the 'Rvertical'-bit is set, the output will be inverted. The range, where the amounts of the inverted offsets are parallel, indicates the stable operating mode of the current sensor. By calculating the sign corrected average of the signals, the overall offset can be decreased. This effect is described below. Figure 11.15 shows also, the less the amount of the bias voltage, the less is the offset of the system. But as mentioned, the noise is increasing at lower voltage, because the signals are smaller as well as the sensitivity at this setup points. Therefore the chip should not be operating close to the bias voltage limits. Thus the ideal range has been decided to a bias code between 15 and 20. In this range, the offset is in an acceptable range and the system supply voltage is not close to the supply limit. To determine the temperature coefficients of the Hall plate, the Hall sensitivity has to be analyzed.

11.2.2 Temperature coefficient and stress compensation

The command bits 'TrimHall_Tc' can be used to adjust the temperature coefficient of the Hall plates. The ratio between the current from the PTAT – circuit and the CTAT circuit shown in Figure 11.16 can be set with the 'TrimHall_Tc'-bits.

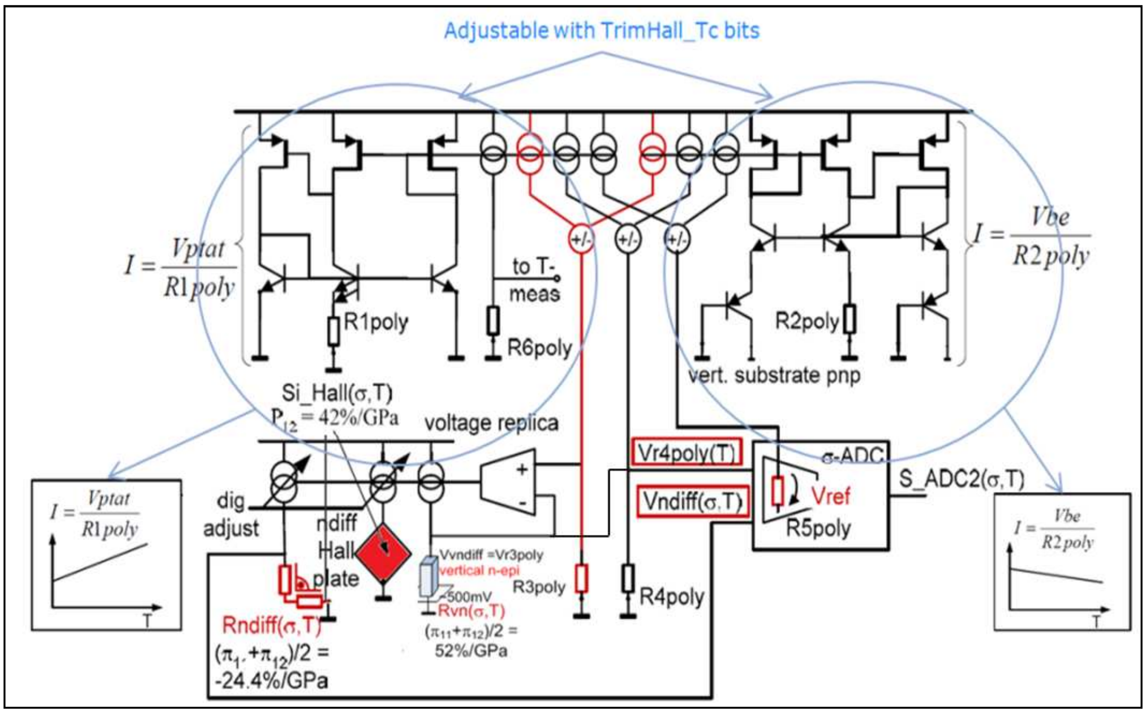


Figure 11.16 : stress compensation and adjustable temperature coefficient [9]

On the top of Figure 11.16 the adjustable current sources are marked. Dependent on the ratio of the current from the 'proportional to absolute temperature' part and the 'complimentary proportional to absolute temperature' part on the right side, the voltage drop at the R3 poly resistor controls the temperature coefficient of the Hall plate. This voltage has also an impact of the stress compensation.

Because of the current mirror at the lower side of Figure 11.16 the current is mirrored to the stress dependent epitaxie resistor. The resistance change due to the mechanical stress does not depend on the direction of the stress tensor at this diffusion resistor. A current mirror, maps the current from the path of the stress dependent resistor to the Hall plate. The correlation between the mechanical stress in GPa and the change of the resistor has a ratio of 52% per GPa. The used Hall plate has also a dependency of the mechanical stress. The change factor of the Hall plate is in the range of 42 percent per GPa. Because of the mirrored current which depends on the resistor value of the vertical stress resistor, an analog pre compensation of the mechanical stress is located at this part of the current sensor. The remaining stress impact, respectively aging effects of the Hall plate (caused by package stress), to the sensitivity of the Hall plate can be compensated in the digital part. Therefore another stress dependent resistor is implemented at the current sensor. Figure 11.16 shows the Rndiff(s,T) resistor in the left corner on the bottom of Figure 11.16. A further current

mirror maps the supply current from the Hall plate into the current path of the diffusion resistor. This current mirror can be set via the interface as described below. The design of the eptitaxi resistor must describe a lateral L-Shape. That means the resistor has to be grouped to the vertical resistor in a perpendicular way. Due to this setup the dependency of the stress tensors is independent to the direction of how the mechanical stress causes to the chip, respectively to the stress sensor.
The different drop voltage at the two stress depending resistors in Figure 11.16 is converted in a digital signal by a one bit continuous time SADC. This signal is send to the output of the current sensor and allows compensating the residual stress error in a further data processing part. Because of the described effect, the setup value of the 'TrimHall_Tc' has also an impact to the stress senor. The top of Figure 11.16 represents the current function which is proportional to the absolute temperature, at the left site. At the right side shows the implemented complimentary proportional to absolutely temperature function. The ratio between I_{ptat} and I_{ctat} current has an impact to the temperature coefficient of the Hall plate as described before.

$$I_{PTAT} = \frac{V_{PTAT}}{R_{1poly}}$$

Equation 11.8 [9]

$$I_{CTAT} = \frac{V_{be}}{R_{2poly}}$$

Equation 11.9 [9]

Dependent on the used Hall bias voltage, respectively at the adjusted bias code, the Hall sensitivity function is shown in Figure 11.17. The used bias code in Figure 11.17 has the value 20 which has been determined due to the stable operating mode of the chip in this range.

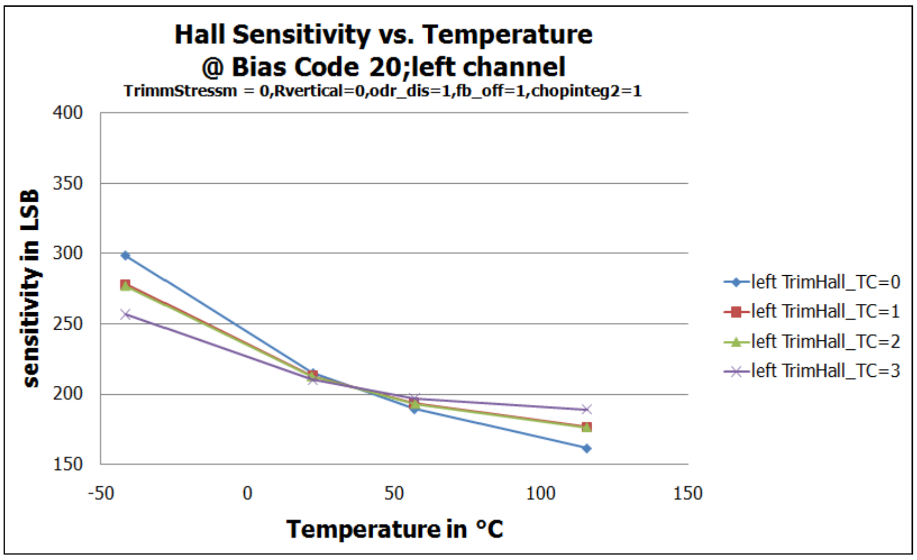


Figure 11.17 : Hall sensitivity versus temperature dependent on temperature coefficent

Figure 11.17 shows the sensitivity for a adjusted bias code 20 dependent on the ambient temperature. The Hall sensitivity is also a function dependent on the adjusted temperature coefficients. The ideal setting for the temperature coefficient represents the ‘TrimHall_Tc’ value, where the sensitivity is as flat as possible. This effect allows a better compensation of the sensitivity function. Figure 11.13 shows that if the bias voltage increases with the ambient temperature, the sensitivity becomes more flat. Because of the direct correlation between the bias voltage and the sensitivity, the gain of the Hall plate becomes plainer if the behavior of the voltage shows a growth at higher temperatures. Figure 11.25 and Figure 11.26 show that the Hall sensitivity increases at lower temperatures. Therefore the temperature coefficient has to be adjusted to pre-compensate this effect.

The current mirror at the left lower side in Figure 11.16, allows the setup of the stress sensor. Therefore the command bit ‘TrimStress’ has to be send to the backend interface. This command allows adjusting the lateral resistor to the reference resistor ‘Rvertical’, respectively to adjust the voltage at these points, which is converted by a SADC in the digital domain.

11.2.3 Offset

The measurements show, that the offset can be improved by inverting the signal after the HADC. The 'Rveritcal'- bit in the backend interface command allows the demodulation of the digital chopping mode by inverting the output of the HADC. Figure 11.18 shows the offset of the Hall plates depending on the 'Rvertical' bit versus the ambient temperature at bias code 20. The temperature coefficient has been set to three because of the mentioned impact to the sensitivity.

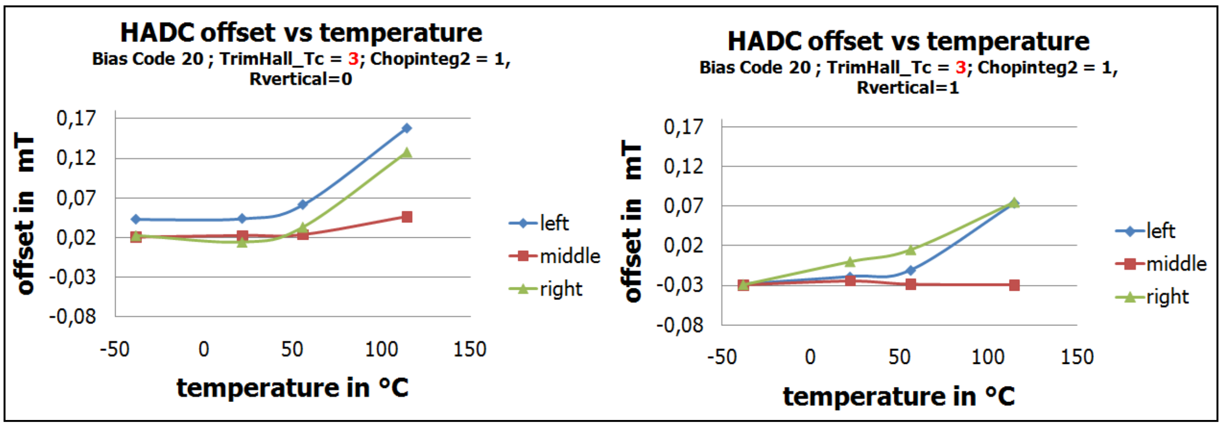


Figure 11.18 : Hall offset versus temperature

By calculating the sign corrected mean value of the inverted offset values, the amount of the offset decreases as shown in Figure 11.19.

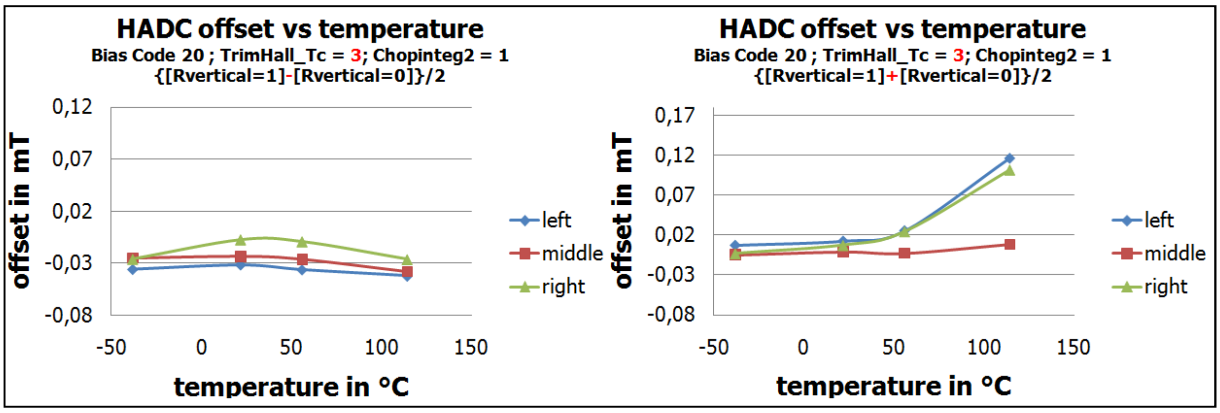


Figure 11.19 : Hall offset versus temperature calculated

The left side of Figure 11.19 shows the offset of the Hall plates by calculating the difference between the inverted offset values and divides them by two. On the right side of Figure 11.19 the signals have been summed and divided by two. Therefore the offset from the HADC can be represented.

$$offset = \frac{\left(offset_{Rverticla=1} - offset_{Rvertical=0}\right)}{2}$$

Equation 11.10

The measurements show that the chopping principle of the HADC causes a smaller offset value.

11.3 Compensation

Due to the fabrication process, every chip has a slightly different performance. Therefore every chip has to be characterized before it can be used to measure current in an efficient way. One reason for the different current sensitivity is that each chip sees different stress components due to its assembly. Dependent on the field of application, stress due to temperature or mechanical forces are influencing each chip. The gain and offset error of the Hall plates are also dependent on the temperature effect. Another cause of different performance is the placement of the Hall sensors on the current path.

Figure 9.4 in the hardware chapter shows the dimension of the PCB and the location where the test chip has to be placed on the PCB. It is important for the sensors performance to place the Hall sensors as exactly as possible on the slits in the PCB. The individual Hall sensitivity has to be determined to allow current measurements with a high accuracy.

To measure the real zero point failure of the sensor, the offset measurements have to be done with a setup where a low magnetic field takes effect on the current sensor. Therefore the chip has to be placed into a 'Mu-metal'-box during the offset measurement. Figure 11.20 shows the setup of the offset measurement.

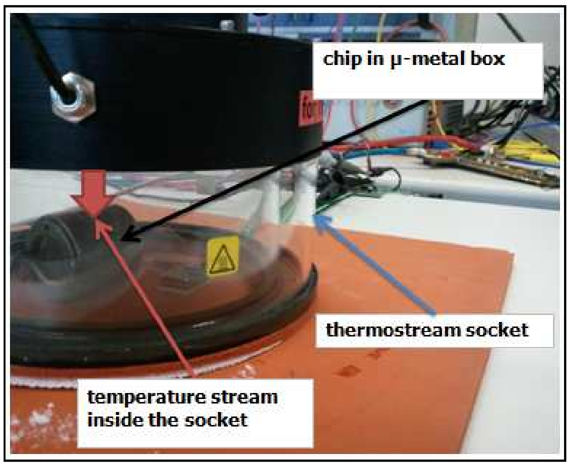


Figure 11.20 : Setup to measure the offset of Hall plates dependent on the temperature

The Mu-metal is a registered trademark of Telcon Metals and is also known as permeability alloy. [20] The box consists of a magnetically soft alloy. This material has a high magnetic permeability which protects the sensor from magnetic fields. Dependent on the bias voltage setting of the chip, the Hall sensitivity and the offset will change. This setup has only been done in the test and evaluating phase of the current sensor to find the ideal settings for the chip. Once the so called standard settings have been ascertained, the calibration for each chip can be done as described below.

As mentioned the offset as well as the sensitivity are functions which are heavily dependent on temperature. To compensate this effect it is essential to know the temperature at each point of measurement. Therefore all calibration measurements have to run over a temperature range which is required in the datasheet respectively in the specification of the current sensor. [11] Dependent on the function of the sensitivity respectively the behavior of the offset versus the temperature range, the number of temperature measure points has to be set. Dependent on the number of measure points, the function can be described by a polynomial. The stress factor is also a result of the effect. But the dependence between the temperature and the stress data has not been analyzed because the stress sensors on the first test chip were not connected correctly. Only a plausibility check of the stress sensors from the redesigned chips has been done.

The goal of the compensation is to find the coefficients to compensate the gain and offset error in a fast and efficient way. The final product should do the compensation algorithm online by using the determined coefficients. Therefore an ISM has to be implemented in the next versions of the current sensors. The process to establish the sensitivity of the chip has been focused on the current measurements. Due to the greater marginal effort to produce the magnetic field with a Helmholtz than with the load current on the copper conductor, the main focus point was the compensation based on a magnetic field, which is produced by the load current on the PCB. Measurements in the Helmholtz coil were only used for the characterizations and basic analyzes of the test chip.

The Equation 11.11 describes how the sensitivity of the Hall plates at the respective temperature set points will be calculated.

$$y_{f(T)} = \frac{dHADC_{LSB(T)}}{dI_{current\_rail_{A}(T)}}$$

Equation 11.11

By knowing the coefficients for the measured temperature ranges, the offset error is given as an initial value of the gain. Figure 11.25 shows the zero point error as a result of the appointed temperatures.

The functions of the three channels are always described as a polynomial higher order. In order to do the compensation at only three temperatures, the offset can only be calculated with a polynomial first or second order. This should be done by calibrating the final product. To ensure an accurate calibration at the final product, the sensitivity compensation has to be done with at least a second order polynomial. Therefore the third order parameter of the chips has to be evaluated in future. This evaluation will be necessary to find a systematical third order coefficient, which can generally be used for all chips. Thus a third order compensation can be done where only a three point calibration has to be done.

11.3.1 Measurements to determine the compensation coefficients

Firstly all chips must be set to a stable state by doing the standard initialization. The ideal settings have been determined in the bias and operating mode chapter as in the chapter 11.1. Therefore the ideal settings for the offset, sensitivity and noise behavior of processed Hall data have been established. The feedback loop has been disabled for these measurements.

After sending the standard settings the cutoff frequency of the two cascaded IIR low pass filters are set by sending the corresponding command to the FPGA. After initialization of the measurement instruments via GPIB, the first temperature set point has to be determined. Due to the volume of the temperature chamber and the massive copper conductor, the temperature has to be set to a higher value, to reach the requested set point inside the chamber. The Figure 11.21 shows the timing setup to reach a stable temperature in the chamber. It shows the settling time between the adjusted temperatures. It takes approximately 19 minutes to stabilize the ambient temperature in the chamber. At this time point the standard deviation of the TADC is smaller than one for a measurement period over one minute. Figure 11.21 also shows the temperature settling time of the self heating effect if the chamber has already reached a stable ambient temperature. The settling time for six degrees over-temperature to the junction temperature on the chip, caused by a current increasing of 45 amperes, is approximately two minutes.

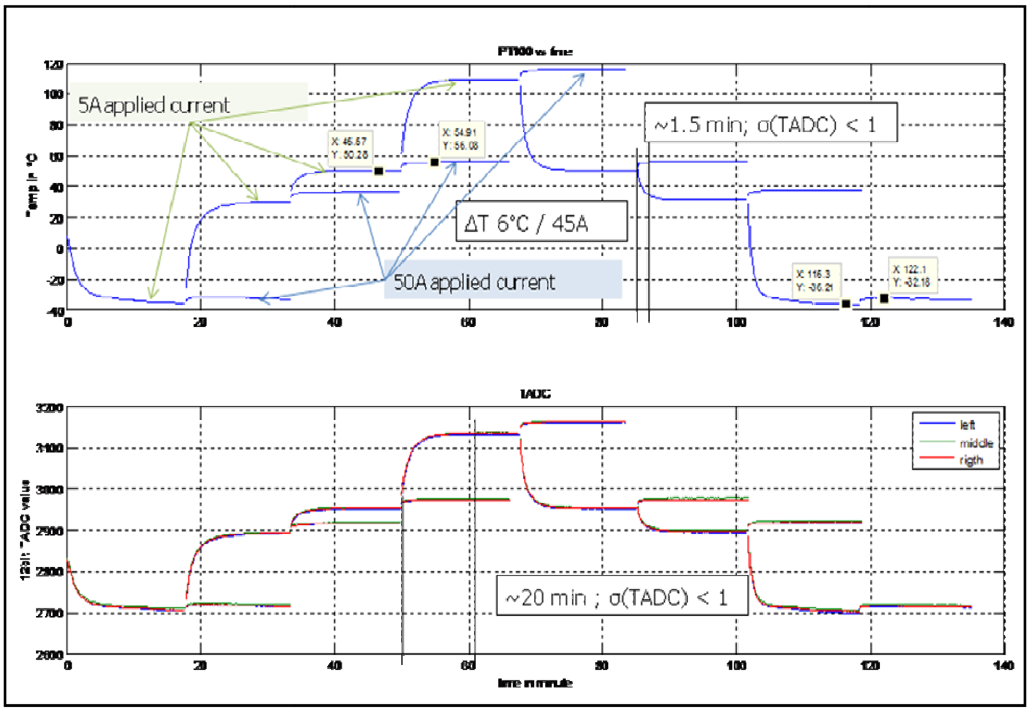


Figure 11.21 : Settling time for a stable temperature in the temperature chamber

After reaching a stable temperature in the thermo chamber, respectively at the current sensor, the temperature value from the PT100 will be read. Accordingly the voltage at the current shunt has to be measured. Following this, the command to read out the data from the FPGA has to be sent. After receiving the requested amount of data from the current sensor, the voltage at the shunt is measured again. The mean value of the two measured results of the shunt will be related to the average value of the HADC data from the FPGA. After changing the polarization of the load current and repeating the measurement cycle, the thermo stream has to be set to the next temperature step.

Equipment for calibration and compensation			
Device	Description	Serial No	Application
PC	PC with USB and GPIB interface and MATLAB		Controls measurement equipment
Measure Hardware	Modulation Circuit		For communication with the chip
FPGA	FPGA with USB board		Signal processing of the CLMCS data and interface to the PC
HP 6681A	0-8V / 0-580A power supply	3640A00452	Current source for CLMCS
Helmholtz coil	Magnetic – Physik MS 210 R = 76Ω k = 0,01405cm 10,9mT/A	109996	Source for homogenous magnetic field
Fluke 8508A	Multimeter	924053362	Multimeter to measure voltage at the shunt in the current path
100A shunt	100A/75mV Rs components	313-788	Shunt to measure the applied current
Agilent E3631A	0-6V,5A / 0+-25V,1A	MY40036959	Controls the relay for the power bridge to change the current polarization
Potter & Brumfield	12V DC relay		Relay to control power bridge
100A Power Bridge	Transistor circuit to change polarization		Changes the current polarization
Agilent E3631A	0-6V,5A / 0+-25V,1A	KR83915076	Power supply for the CLMCS and modulation circuit
K2420	3A Sourcemeter	0660764	To force the Helmholtz coil with the required current
K2000	Multimeter	0792595	To measure voltage on the test pin of the CLMCS
K2000	Multimeter	1015964	To measure voltage at the PT100
Thermojet	FTS		For temperature sweep
Temperature chamber			Chamber for stable ambient temperature

11.3.2 Characterization for the compensation

As noted previously, every channel will be calculated and compensated separately. Figure 11.22 describes how the characterization for each channel is done. The coefficient for the compensation can be derived from this measurement.

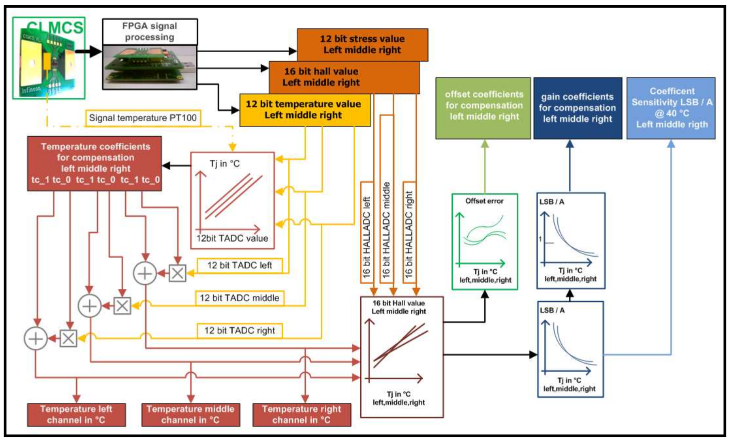

CLMCS
FPGA signal processing
12 bit stress value Left middle right
16 bit hall value Left middle right
12 bit temperature value Left middle right
Signal temperature PT100
Temperature coefficients for compensation left middle right tc_1 tc_0 tc_1 tc_0 tc_1 tc_0
Tj in °C
12bit TADC value
12 bit TADC left
12 bit TADC middle
12 bit TADC right
16 bit HALLADC left
16 bit HALLADC middle
16 bit HALLADC right
16 bit Hall value Left middle right
Tj in °C left,middle,right
offset coefficients for compensation left middle right
gain coefficients for compensation left middle right
Coefficent Sensitivity LSB / A @ 40 °C Left middle rigth
Offset error
LSB / A
1
Temperature left channel in °C
Temperature middle channel in °C
Temperature right channel in °C

Figure 11.22 : Characterization to find coefficients for compensation

The first coefficients which have to be calculated are the temperature coefficients tc_ɑ and tc_β for each channel. If these coefficients are known it is possible to make conclusions about the actual junction temperature in degrees Celsius on the chip. Therefore a correlation between the TADC values from the chip and the measured temperature value from the PT100 sensor has to be found. Figure 11.23 shows the TADC value compared to the temperature.

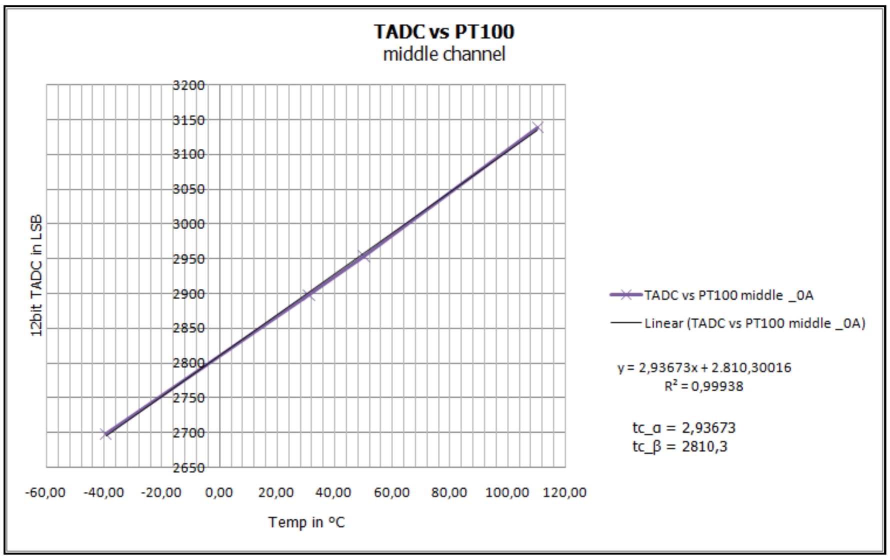


Figure 11.23 : TADC compared to temperature

The x-axis shows the mean value of the PT100 temperature from each period where the gain and the offset has been measured. The y-axis shows the corresponding mean values of the values from the temperature ADCs. The TADC value will be described by a linear function.

Therefore a linear polynomial fit is done to determine the coefficients for the temperature calculation.

In the next step the junction temperature of each channel will be calculated using the determined coefficients.

$$T_j = \frac{TADC - tc_\beta}{tc_\alpha}.$$

Equation 11.12

It is also possible to calculate the gain and offset error relating to the twelve bit TADC value, but to make the calculation more demonstrative the TADC value will be transformed into degrees Celsius by using the temperature coefficients. In the final product the temperature will not be calculated for the compensation. Only the ADC values from the chip will be used for the compensation to avoid unnecessary calculation which may increase the failure due to roundoff errors.

For the compensation it is important to ensure that the temperature of the system has reached a stable state. Therefore the TADC value will be logged because the standard deviation of a measure period of two minutes will be smaller than one. If this status has been reached, all other ADC values will be saved and used for further calculations. For the further calculations, an average value of the data from the TADC and HADC will be used. For the compensations which will be described below the average of 400 sampled ADC values has been used for every measure point.

Figure 11.24 shows the temperature profile during the calibration.

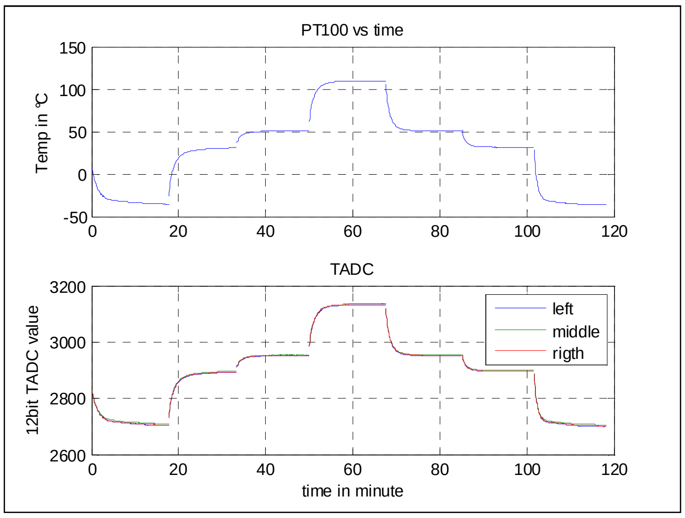


Figure 11.24 : Temperature profile during calibration

In the next step the sensitivity of the Hall probe has to be calculated for each settled temperature period. It is known that the relation between magnetic field and the Hall voltage is linear.

$$U_H = A_H \frac{I \cdot B}{d}$$

Equation 11.13 [4]

Therefore a linear fit can also be done to calculate the Hall gain of the chip. Figure 11.25 shows the sensitivity from each channel at the individual temperature set points in LSB per ampere.

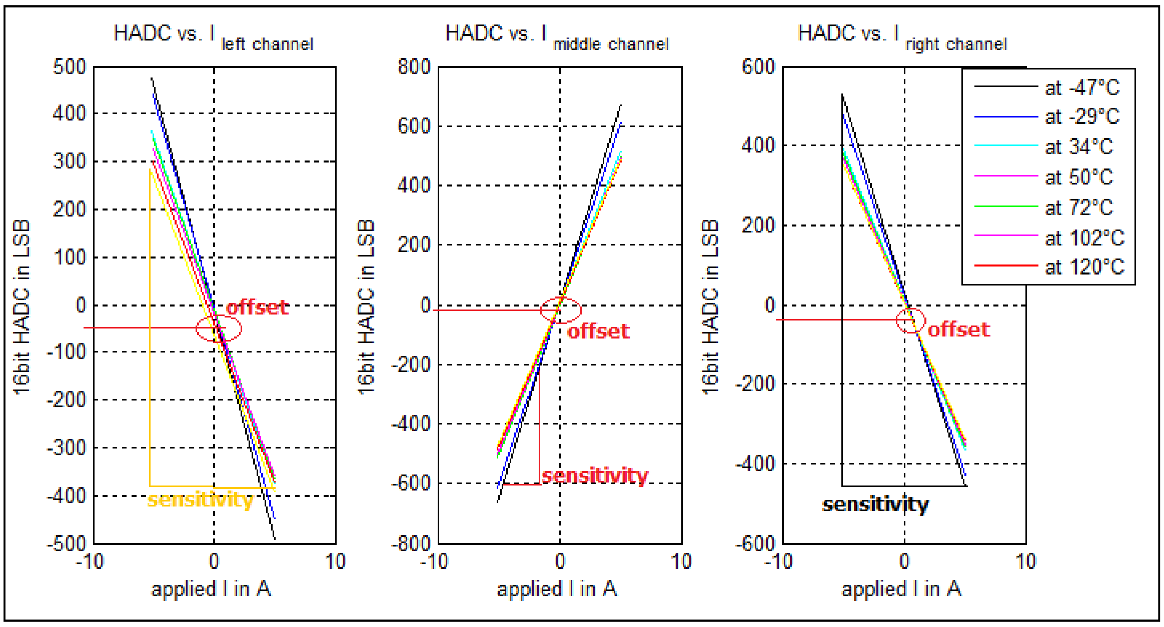


Figure 11.25 : Hall sensitivity dependent on the temperature

Figure 11.25 delivers the Hall sensitivity of each channel at every temperature set point.

$$H_{sensitivity(T)} = \frac{dHADC_{(T)}}{dI_{(T)}}$$

Equation 11.14

$$HADC_{(T)} = H_{sensitivity(T)} \cdot I + offset_{(T)}$$

Equation 11.15

By using the linear Equation 11.14 the offset will be the intercept at zero amperes. Therefore the zero point error, respectively the offset can be calculated as follows.

$$offset_{(T)} = HADC_{(T)} - \frac{dHADC_{(T)} \cdot I}{dI_{(T)}}$$

Equation 11.16

Figure 11.26 shows the calculated sensitivities of all three channels versus the appointed temperature.

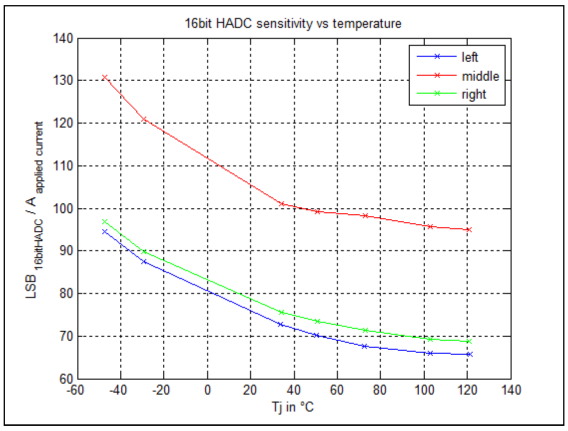


Figure 11.26 : Hall sensitivity of each channel dependent on the temperature

The sensitivity of the middle channel is always higher than the sensitivity of the other two channels because of the higher current density at the middle slit. Figure 11.26 shows clearly how the sensitivity is decreasing at each Hall probe when the temperature increases. To compensate this effect, the inverse function for every channel has to be identified. This has to be done once for each chip. Therefore it is important that the function in Figure 11.26 is reproducible, because the total Hall–sensitivity–compensation is based on this function.

$$y_{LSB/A} = f_{(temperature)}$$

Equation 11.17

Figure 11.27 shows the behavior of the gain versus the temperature after normalizing the function with the temperature value to the nearest value of 40 degrees Celsius.

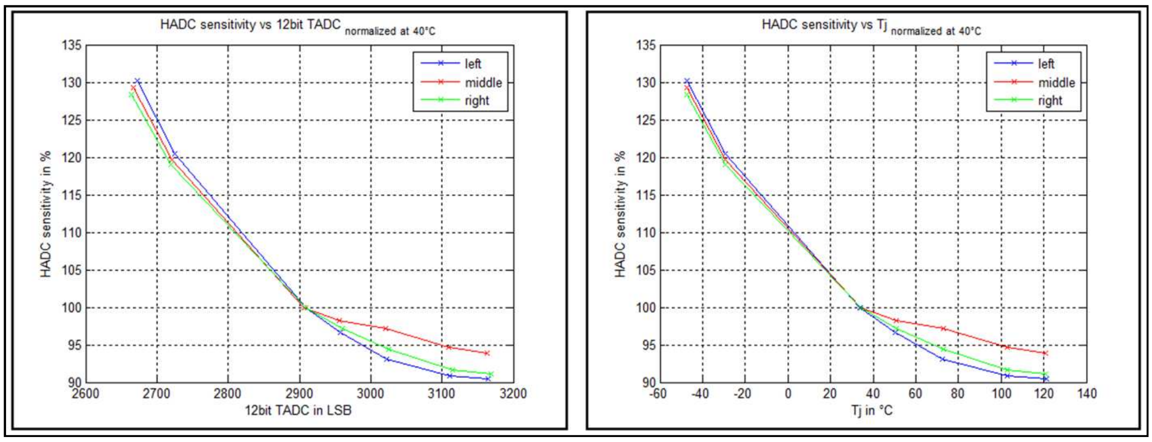


Figure 11.27 : Hall sensitivity normalized at 40°C dependent on TADC and Tj

After normalizing the function a polynomial fit fourth order has to be done to get the characterizing coefficients st_c. Therefore the TADC value has to be transformed in another range, so that the fitting coefficients will have a higher amount allowing compensation with more accuracy. Thus, the 12bit TADC value will be increased by the factor 2^{11} and divided by the factor 1000.

$$TADC_{\_fit} = \frac{TADC - 2^{11}}{1024}$$

Equation 11.18

When performing the compensation it is necessary to transform the TADC values into the same range, before the calculations can be done.

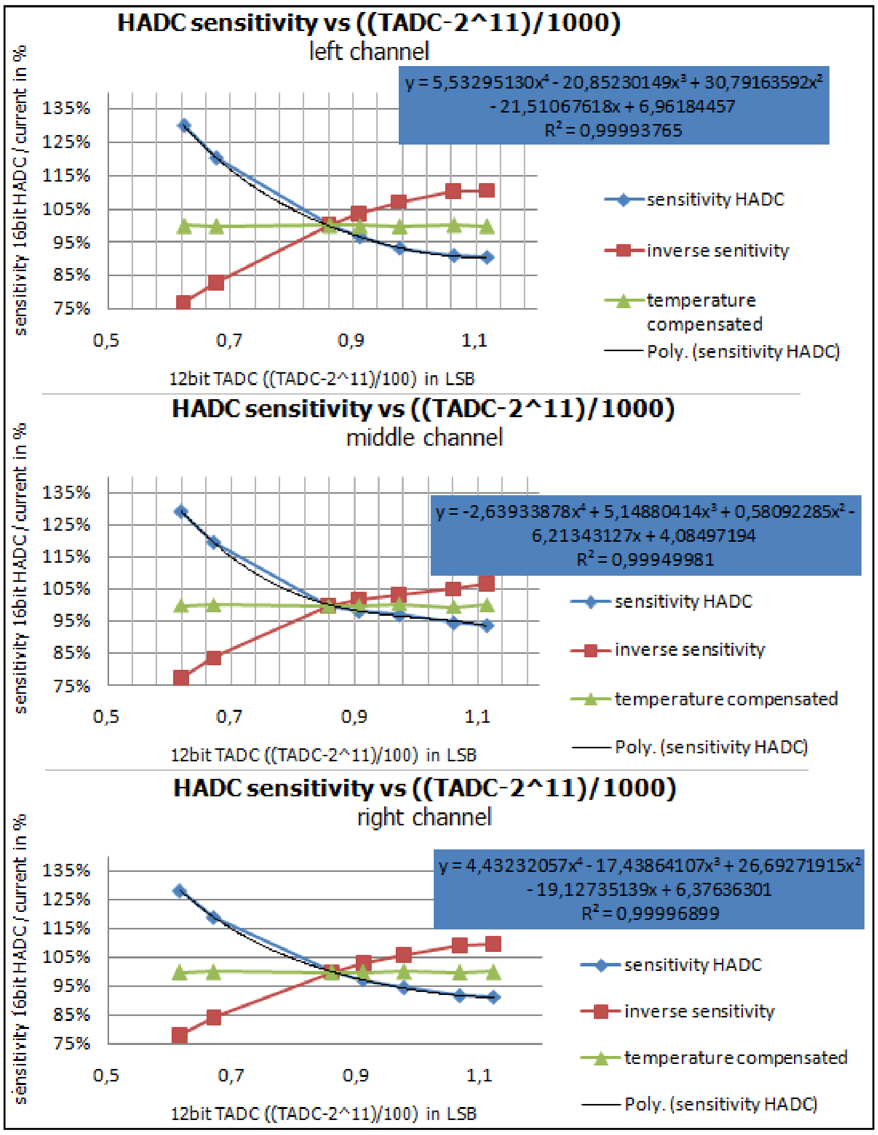


Figure 11.28 : Normalized Hall sensitivity in percent

Figure 11.28 shows the normalized sensitivity versus the *TADC_fit* value. The polynominal fit has been done by the root mean square method. The formulas in the figures are representing the coefficents required to calculate the sensitivity. By multiplying the blue Hall sensitivity function with the recalculated inverse function which is represented in red, the sensitivity has been compensated. The temperature compensated function is shown in green at the Figure 11.28.

This figure shows clearly that the sensitivity will be stable at around 100 percent for the whole temperature range.

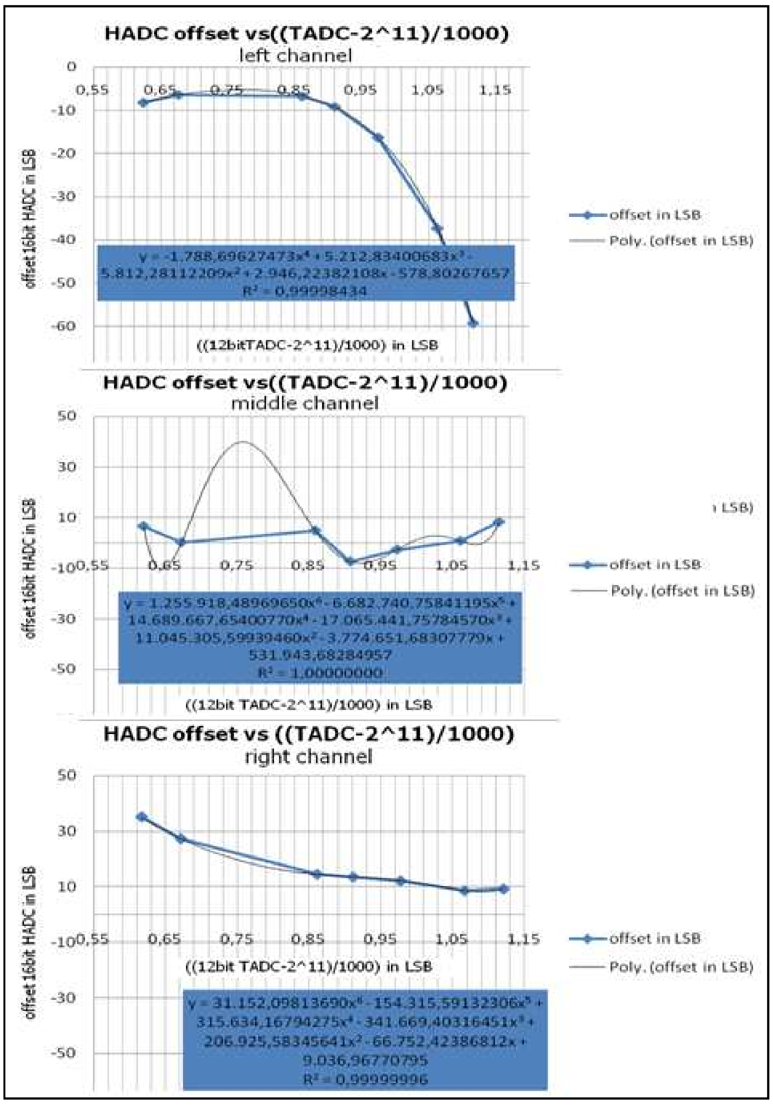


Figure 11.29 : HADC offset depending on the temperature

The amount of the intercepts from the gain calculation before is shown in Figure 11.29 as a function dependent on the temperature. A transformation of the TADC to a smaller value range will also deliver better fit coefficients for the compensation. In the same way as before the TADC value will be increased by the factor 2^{11}, which corresponds to the half modulation range of the TADC. Afterwards the result will be divided by the factor 1024 before the polynomial fit will be done. The factors for the current sensitivities for each channel will be the gain at the temperature value where the normalization of the sensitivity belongs.

$$sens_{left} = \left. \frac{dHADC_{left}}{dI_{current\_rail}} \right|_{at\_normalized\_temperature(40°C)}$$

Equation 11.19

$$sens_{middle} = \left. \frac{dHADC_{middle}}{dI_{current\_rail}} \right|_{at\_normalized\_temperature(40°C)}$$

Equation 11.20

$$sens_{right} = \left. \frac{dHADC_{right}}{dI_{current\_rail}} \right|_{at\_normalized\_temperature(40°C)}$$

Equation 11.21

Figure 11.25 shows how the Hall sensitivity is reactive to variations in temperature. Figure 11.30 shows the correlation between temperature changing and the loss of sensitivity in percent.

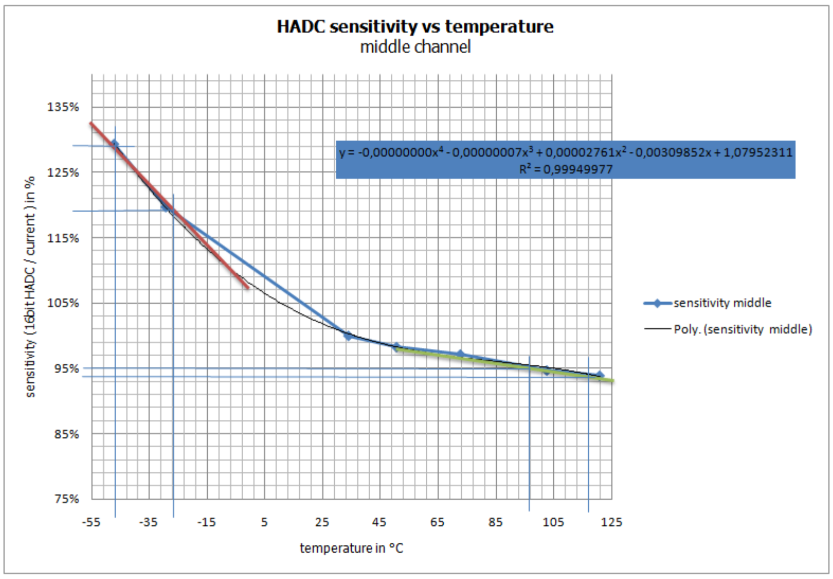


Figure 11.30 : Hall sensitivity in percent depending on the temperature

Therefore the temperature has to be very stable during the compensation to guarantee an accurate calculation. Figure 11.30 shows the sensitivity of the middle channel versus the temperature range. In the range from minus 40 to minus 20 degrees a change of 20 degrees causes a reduction in sensitivity of 10 percent. That means that an error of 1°C during the compensation causes an error of a half percent for the current calculation.

$$error_{calc\_current} = \left. \frac{dsensitivity}{dT_j} \right|_{partwise}$$

Equation 11.22

It is again important to guarantee a very stable temperature during the compensation measurements. At the first compensation routines the applied current was swept through from minus 10 to plus 10 amperes. This function of the current causes a temperature change during the compensation as shown in Figure 11.31.

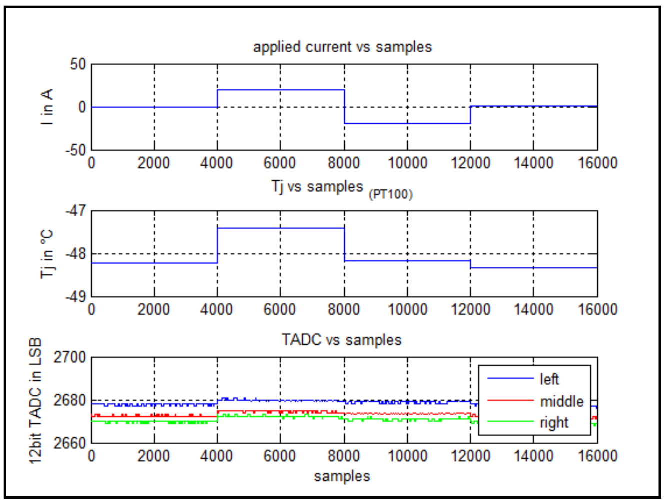


Figure 11.31 : Temperature change at calibration due to the calibration current

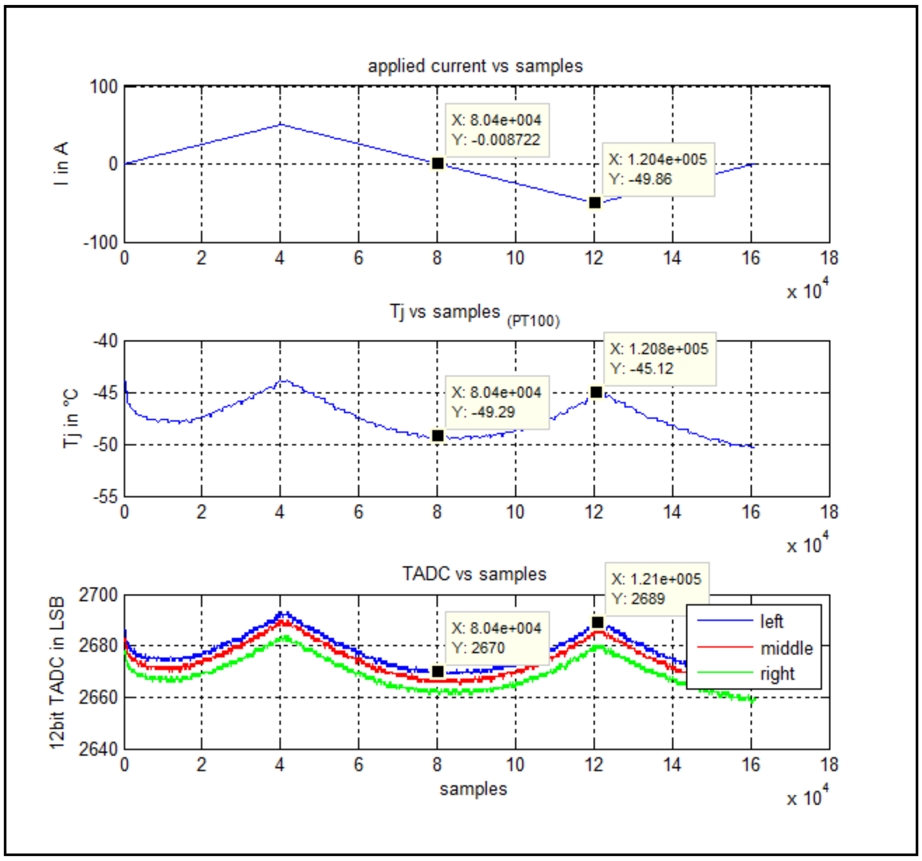


Figure 11.32 : temperature change depending on the applied current

Figure 11.32 shows the behavior of the temperature dependent on the self-heating effect. This drawing shows that a current of 50 A causes a self heating effect of at least 4°C. Figure 11.31 shows a heating of 1 °C if current changes from zero to ten amperes. As described, every current-sensitivity of the Hall probes has to be dedicated to a temperature to achieve the individual sensitivity functions of the sensors. To avoid errors due to this inaccuracy, the chip has to be forced with the same current amount during the compensation process. Only the polarization has to be changed to determine the gain.

11.3.3 Compensation

Figure 11.33 describes how the current calculation will be done by using the compensation coefficients. This figure describes sensitivity and offset compensation to the 3rd order.

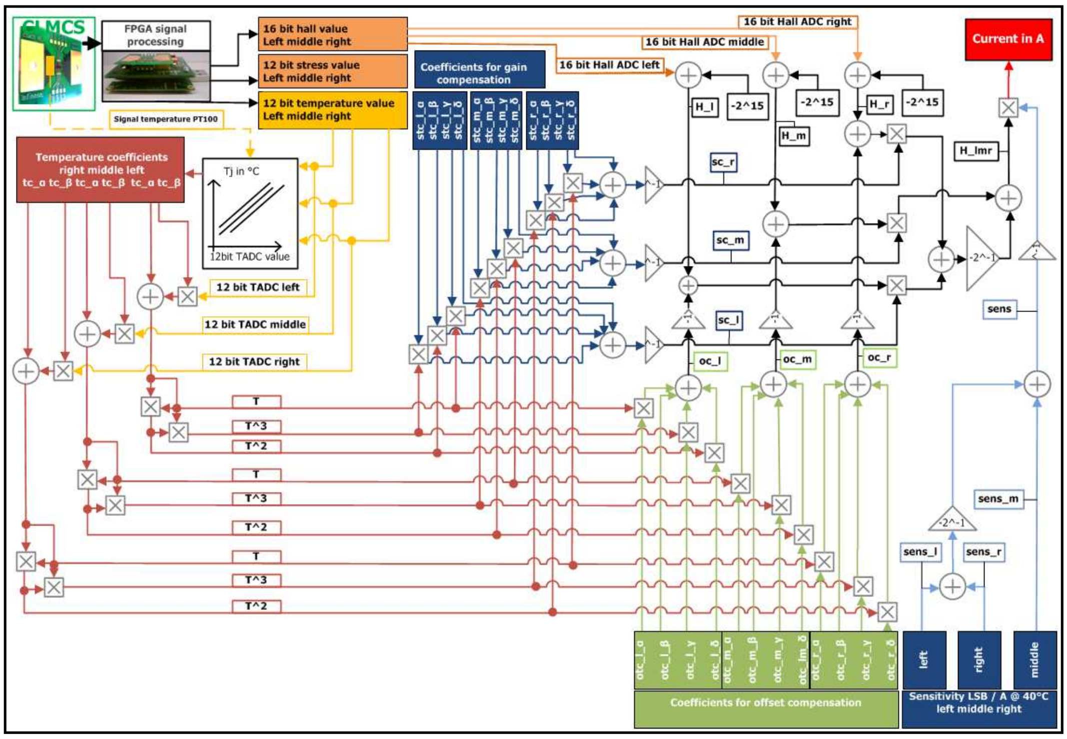

FPGA signal processing
16 bit hall value Left middle right
12 bit stress value Left middle right
12 bit temperature value Left middle right
Signal temperature PT100
Coefficients for gain compensation
16 bit Hall ADC right
16 bit Hall ADC middle
16 bit Hall ADC left
Current in A
Temperature coefficients right middle left tc_α tc_β tc_α tc_β tc_α tc_β
Tj in °C
12bit TADC value
12 bit TADC left
12 bit TADC middle
12 bit TADC right
H_l
H_m
H_r
H_lmr
-2^15
sc_r
sc_m
sc_l
oc_l
oc_m
oc_r
sens
sens_m
sens_l
sens_r
T
T^3
T^2
Coefficients for offset compensation
Sensitivity LSB / A @ 40°C left middle right
left
right
middle

Figure 11.33 : Compensation with determined coefficients from the calibration

For the first measurements in the laboratory, the polynomial fit for the offset was done to the 6th order to ensure an exact fit for all seven temperature points. The sensitivity fit was done to the order of four which is also higher as described in Figure 11.33. First of all the value 2^{15} will be subtracted from each Hall value of every channel. Because the Hall signal is a 16 bit data stream in a unsigned format. By subtracting the half of the range of values the zero-point corresponds to zero.

$$16bitHADC_{left\_middle\_right} = HADC - 2^{15}$$

Equation 11.23

After transforming the TADC values in, the offset can be calculated for each channel using Equation 11.24 to Equation 11.26.

$$oc_l = \left(\frac{TADC_{left} - 2^{11}}{1024}\right)^3 \cdot otc_{l_\alpha} + \left(\frac{TADC_{left} - 2^{11}}{1024}\right)^2 \cdot otc_{l_\beta} + \left(\frac{TADC_{left} - 2^{11}}{1024}\right) \cdot otc_{l_\chi} + otc_{l_\delta}$$

Equation 11.24

$$oc_m = \left(\frac{TADC_{middle} - 2^{11}}{1024}\right)^3 \cdot otc_{m_\alpha} + \left(\frac{TADC_{middle} - 2^{11}}{1024}\right)^2 \cdot otc_{m_\beta} + \left(\frac{TADC_{middle} - 2^{11}}{1024}\right) \cdot otc_{m_\chi} + otc_{m_\delta}$$

Equation 11.25

$$oc_r = \left(\frac{TADC_{rightt} - 2^{11}}{1024}\right)^3 \cdot otc_{r_\alpha} + \left(\frac{TADC_{right} - 2^{11}}{1024}\right)^2 \cdot otc_{r_\beta} + \left(\frac{TADC_{right} - 2^{11}}{1024}\right) \cdot otc_{r_\chi} + otc_{r_\delta}$$

Equation 11.26

Equation 11.24 too Equation 11.26 describe how the offset has to be calculated.

This value has to be multiplied with the inverse Hall sensitivity function which is described by the TADC values and the compensation coefficients 'st_c'.

$$sc_l = \frac{1}{\left(\frac{TADC_{left} - 2^{11}}{1024}\right)^3 \cdot stc_{l_\alpha} + \left(\frac{TADC_{left} - 2^{11}}{1024}\right)^2 \cdot stc_{l_\beta} + \left(\frac{TADC_{left} - 2^{11}}{1024}\right) \cdot stc_{l_\chi} + stc_{l_\delta}}$$

Equation 11.27

$$sc_m = \frac{1}{\left(\frac{TADC_{middle} - 2^{11}}{1024}\right)^3 \cdot stc_{m_\alpha} + \left(\frac{TADC_{middle} - 2^{11}}{1024}\right)^2 \cdot stc_{m_\beta} + \left(\frac{TADC_{middle} - 2^{11}}{1024}\right) \cdot stc_{m_\chi} + stc_{m_\delta}}$$

Equation 11.28

$$sc_r = \frac{1}{\left(\frac{TADC_{right} - 2^{11}}{1024}\right)^3 \cdot stc_{r_\alpha} + \left(\frac{TADC_{right} - 2^{11}}{1024}\right)^2 \cdot stc_{r_\beta} + \left(\frac{TADC_{right} - 2^{11}}{1024}\right) \cdot stc_{r_\chi} + stc_{r_\delta}}$$

Equation 11.29

Equation 11.27, Equation 11.28 and Equation 11.29 describe how to calculate the inverse sensitivity function for each channel.

$$H_l = HADC_{left} - 2^{15}$$

Equation 11.30

$$H_m = HADC_{middle} - 2^{15}$$

Equation 11.31

$$H_r = HADC_{right} - 2^{15}$$

Equation 11.32

Equation 11.33 describes how to calculate the offset and sensitivity compensated Hall value for all three channels. The temperature compensated value has to be divided by the common sensitivity factor `sens' which is described in Equation 11.34.

$$H_{lmr} = ((H_m - oc_m) \cdot sc_m) - \left(\frac{((H_l - oc_l) \cdot sc_l) + ((H_r - oc_r) \cdot sc_r)}{2}\right)$$

Equation 11.33

$$sens = sens_m - \frac{sens_l + sens_r}{2}$$

Equation 11.34

$$current = \frac{H_{lmr}}{sens}$$

Equation 11.35

To achieve a definite and accurate correlation of the current value, respectively of the magnetic field, the outcome has to be divided by the common sensitivity of the channels as described in Equation 11.35.

Figure 11.34 shows the applied current and the uncompensated amount of the calculated current from the chip as well as the offset compensated current from the test chip.

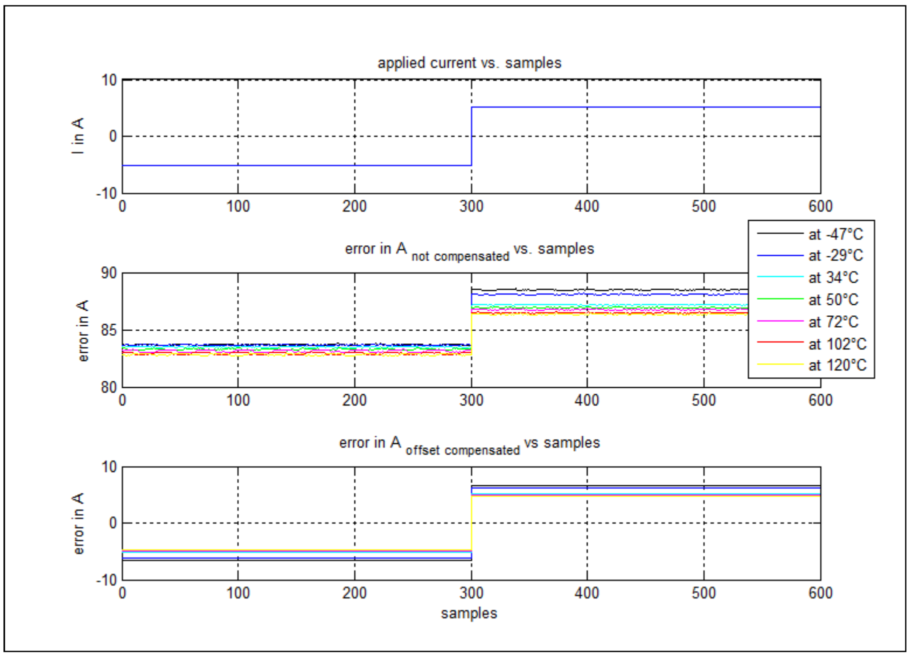


Figure 11.34 : Applied current and error in amp

Figure 11.35 shows the offset and sensitivity temperature compensated error in amp for each channel separately.

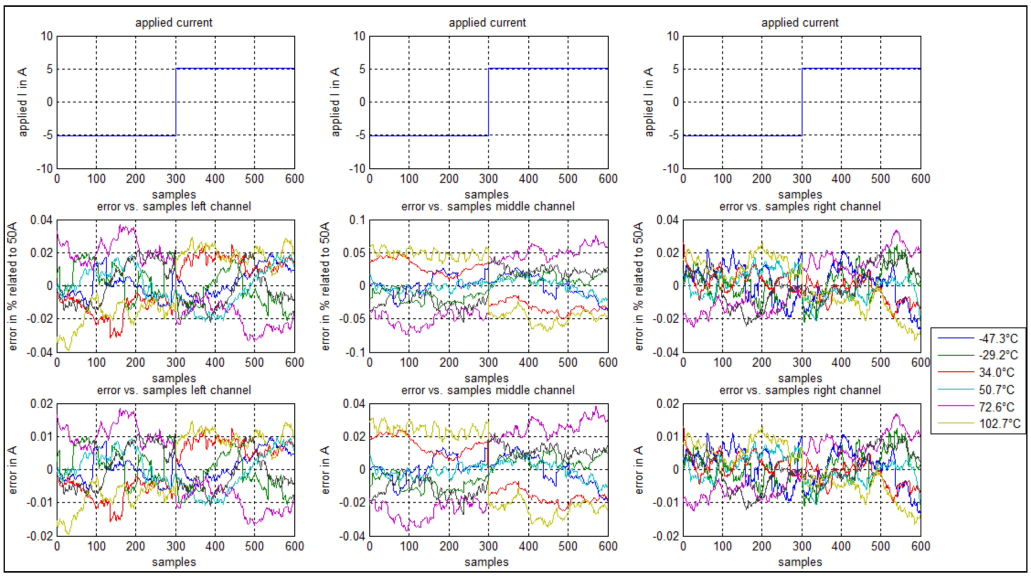


Figure 11.35 : Error in A and % after temperature compensation for each channel

Figure 11.36 shows the error of the temperature compensated current of the testchip in ampere.

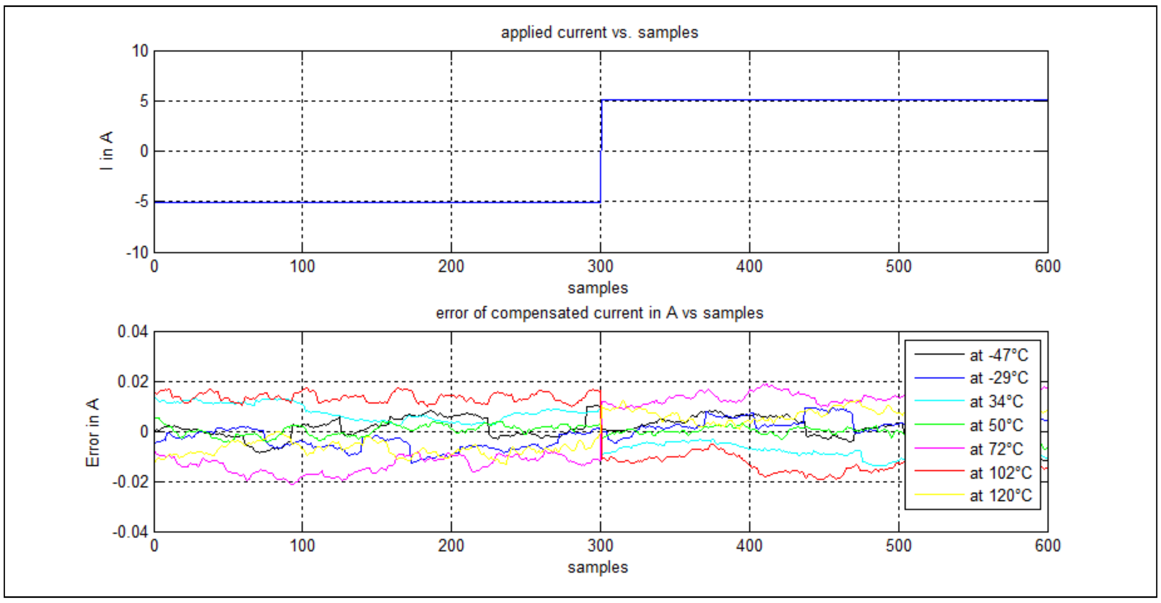


Figure 11.36 : Overall error in ampere after temperature compensation

This compensation has been done with the same data which has been used for the current calculation. In the final product the compensation should be done only once and the out coming coefficients from this compensation should be used for all further current calculations with this chip. Therefore another chip has been compensated in the laboratory. The applied current for the compensation has been done with five ampere. The determined compensation coefficients have to be used to calculate the current which will be applied after the compensation at different temperatures. Figure 11.37 shows the current function, which has been applied at the same temperature ranges as the calibration. The chip was exposed to an additional temperature gradient which occurs due to the self-heating effect of the current. Figure 11.32 shows the behavior of the temperature on the chip during one measure cycle. The changing of the temperature does not have to be a problem after the compensation. The bottom of Figure 11.37 shows the error of the compensated current in amperes.

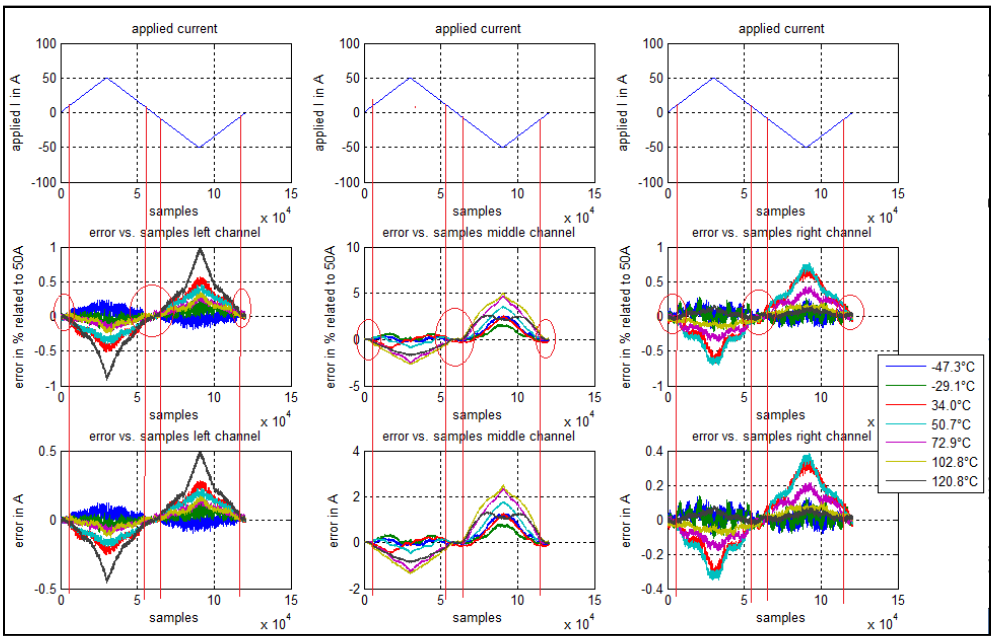


Figure 11.37 : Error in amps after compensation with coefficients from the 5A calibriaton

The Figure above shows, that the compensation was only successful in three small intervals which are highlighted in red. More precisely, the compensation has been successful in the same current range as it has been in the previous calibration. This illustrates that the behavior of the chip is not the same in a situation where five or where fifty amperes are applied. The effects are non linear, which can be explained by stress effects on the chip. The stress effects cannot be compensated with the first test chip, because the stress sensors of these devices are not connected. The presumption of a non linear effect based on the Hall Effect in the copper conductor at higher current has been eliminated by simulations. Simulations have shown that a non linear effect, occurring at a very high current, will only take place at a very high current amount in a range of *kA*.

If the compensation would be done with 50 amps the error of the compensated current would not be as high as if the compensation would be done with 5 amperes. Figure 11.38 shows the error in ampere and in percent, related to 50 ampere for each channel separate if the calibration has been done with 50 amps.

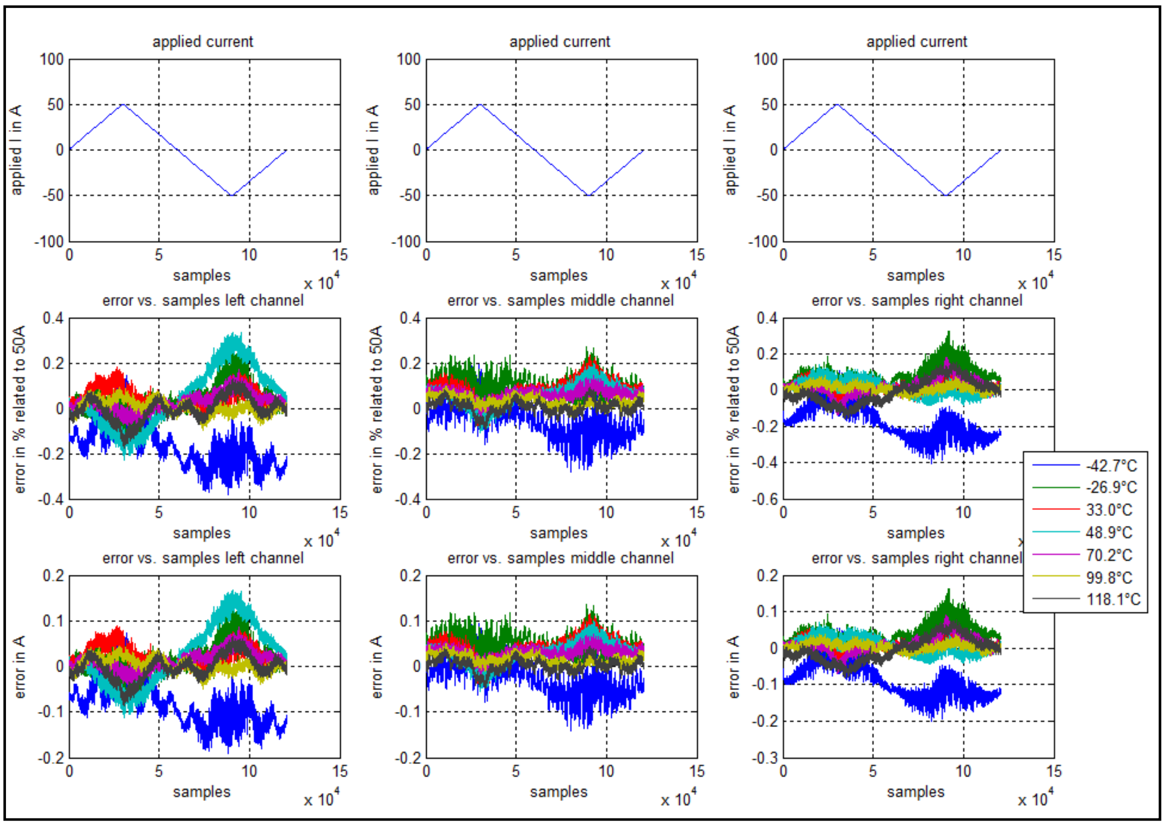


Figure 11.38 : Error after the compensation with coefficients from the 50A calibriation

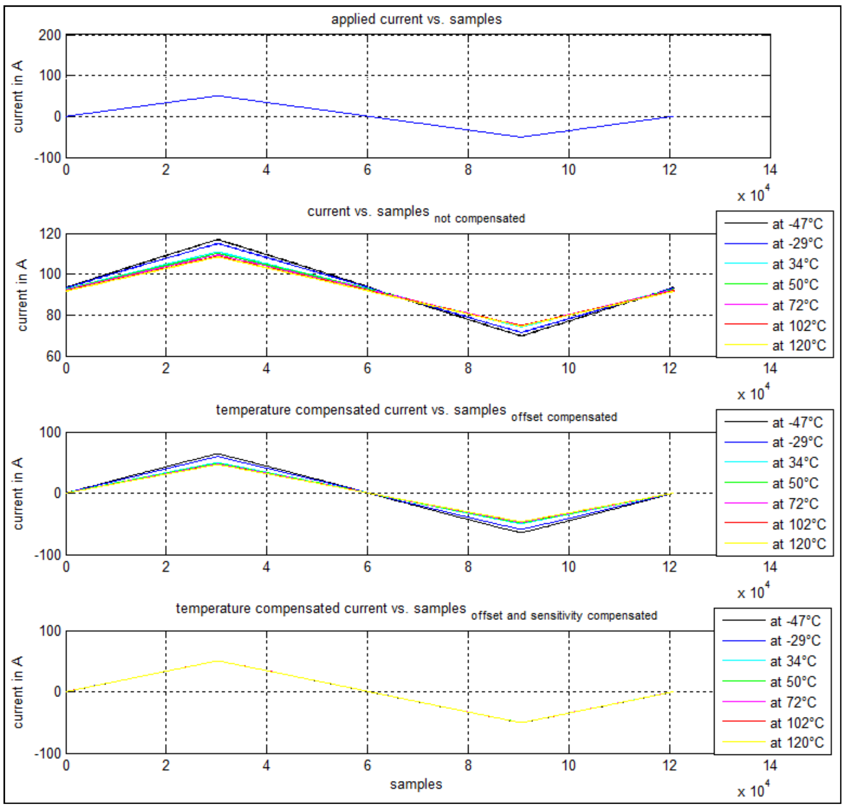


Figure 11.39 : Applied current and compensated as uncompensated current

Figure 11.39 shows the applied current versus the samples in the top drawing. The second drawing in Figure 11.39 shows the uncompensated current. The third drawing describes the offset compensated current. The gain error which is represented in the third drawing is shown as compensated in drawing four. The overall error in amperes is shown in Figure 11.40.

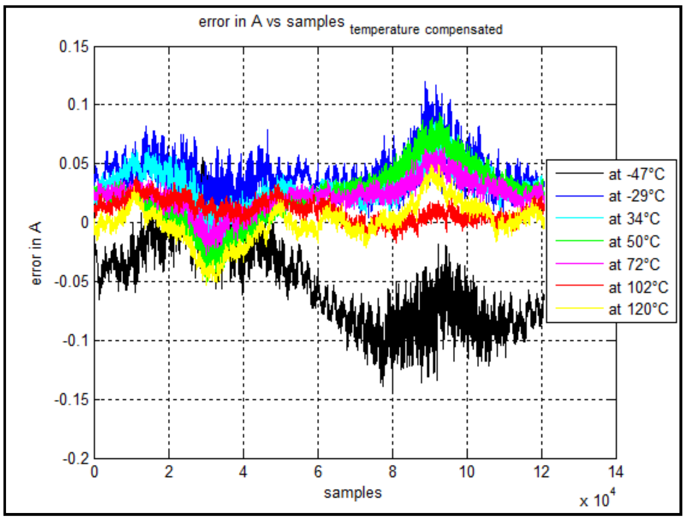


Figure 11.40 : Overall error in amps after temperature compensation

11.3.4 Analyzing the non linear effect of the test chip

To analyze the non linear effect, the following measurement must be done. The Hall sensitivity dependent on the temperature has to be determined in different ways. The setup of the measure is described in Figure 11.41.

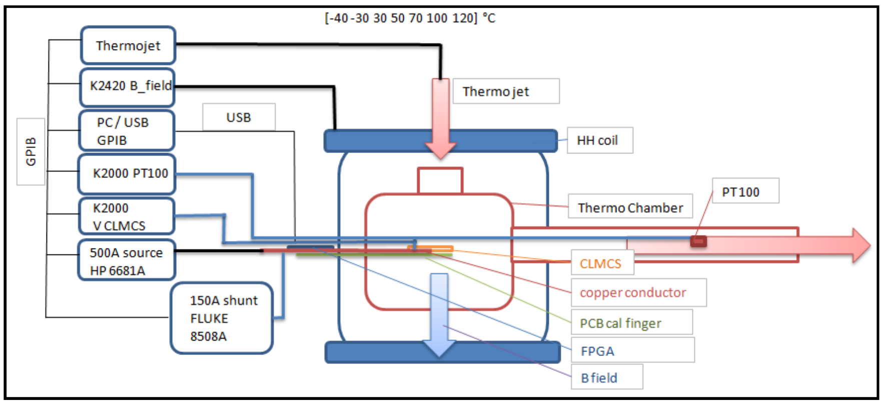


Figure 11.41 : Measure setup to analyze the non linear effect

To establish the Hall sensitivity in LSB/mT, a magnetic field in the z direction has been applied. The field has been generated with a Helmholtz coil. The thermo chamber is placed inside the Helmholtz coil. Figure 9.30 shows how the measure setup has been done. The current sensor is placed in the centre of the Helmholtz, inside the thermo chamber. The calibration finger, which is shown in Figure 9.31, connects the CLMCS with the FPGA and allows the current to flow to the CLMCS via two copper conductors. To analyze the behavior of the chip at different amounts of current, the chip has to be forced with different amounts of current. The Hall sensitivity of the sensor has to be established by varying the magnetic field, which is produced by the Helmholtz coil. This field will be superposed to the magnetic field dependent on the adjusted load current. Also the Hall sensitivity in LSB/A has to be determined by varying the load current. Figure 11.42 gives an overview of how the measurement has been done.

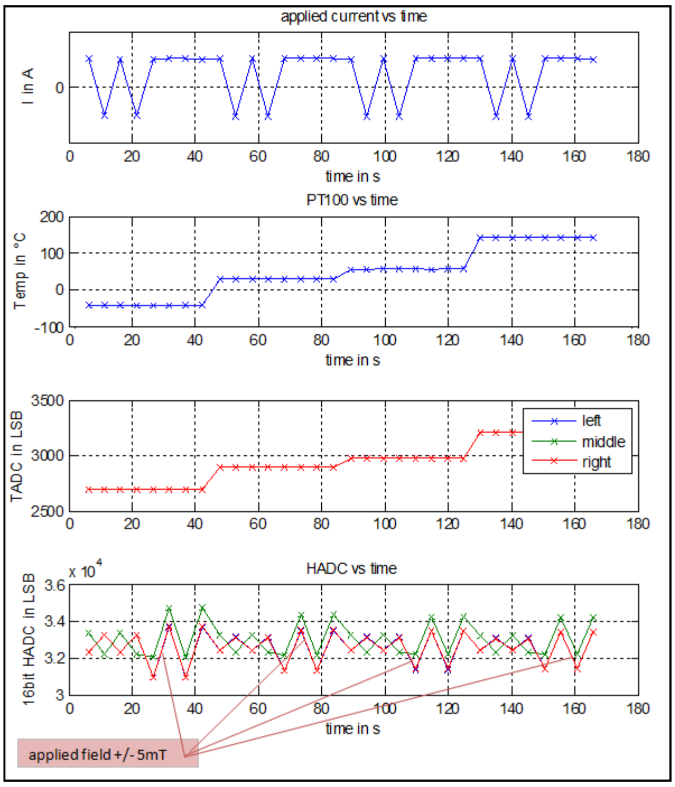


Figure 11.42 : Flow of measuring the non linear effect

At every temperature set point the chip will be forced with a constant load current. To determine the Hall sensitivity in LSB/A, the polarization of the current will be changed. Afterwards a magnetic field, produced by the Helmholtz coil will be superposed to the magnetic field which depends on the load current. To establish the sensitivity, the magnetic field has to be applied in a positive as well as in a negative direction. After repeating these measurements for every temperature set point, two further temperature sweeps must be done. The subsequent measurements are distinguished from each other by the amount of the load current. After the three measured cycles, one with zero ampere load current, one with three and another with 50 amperes, the results will be compared. Figure 11.43, Figure 11.44 and Figure 11.45 are showing that the Hall sensitivities versus the temperature are different.

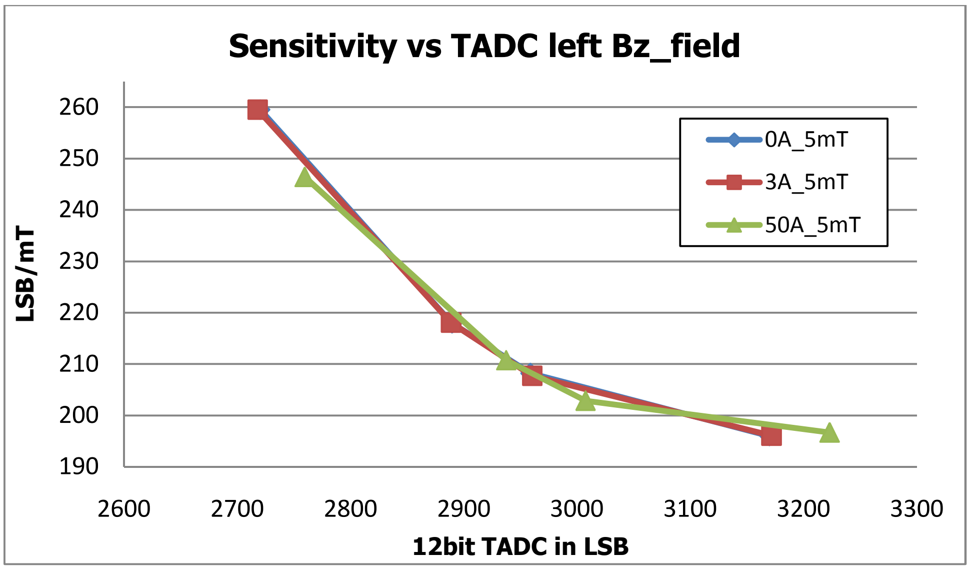


Figure 11.43 : Different Hall sensitivity depending on the load current at the left channel

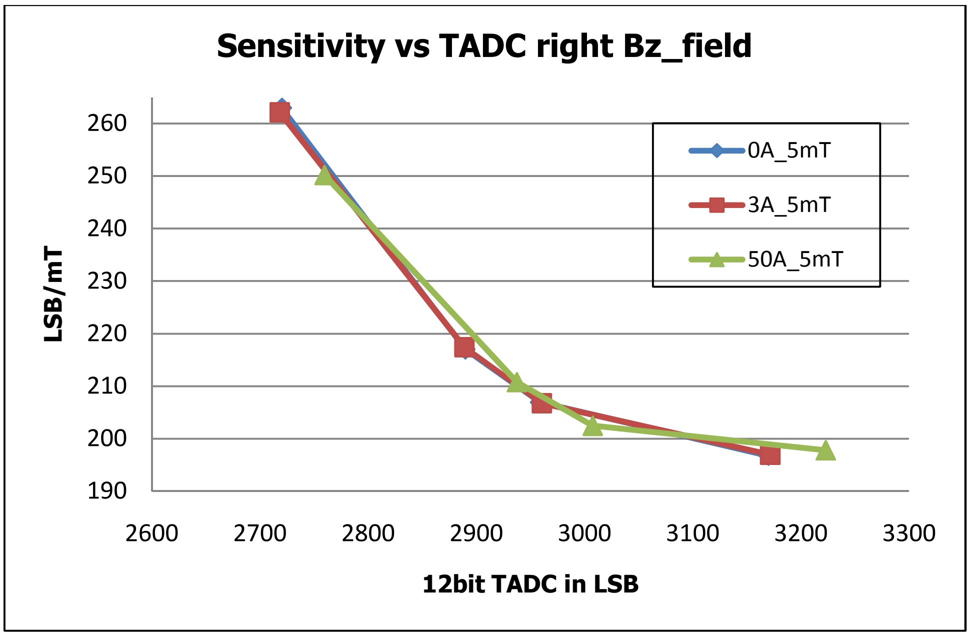


Figure 11.44 : Different Hall sensitivity depending on the load current at the right channel

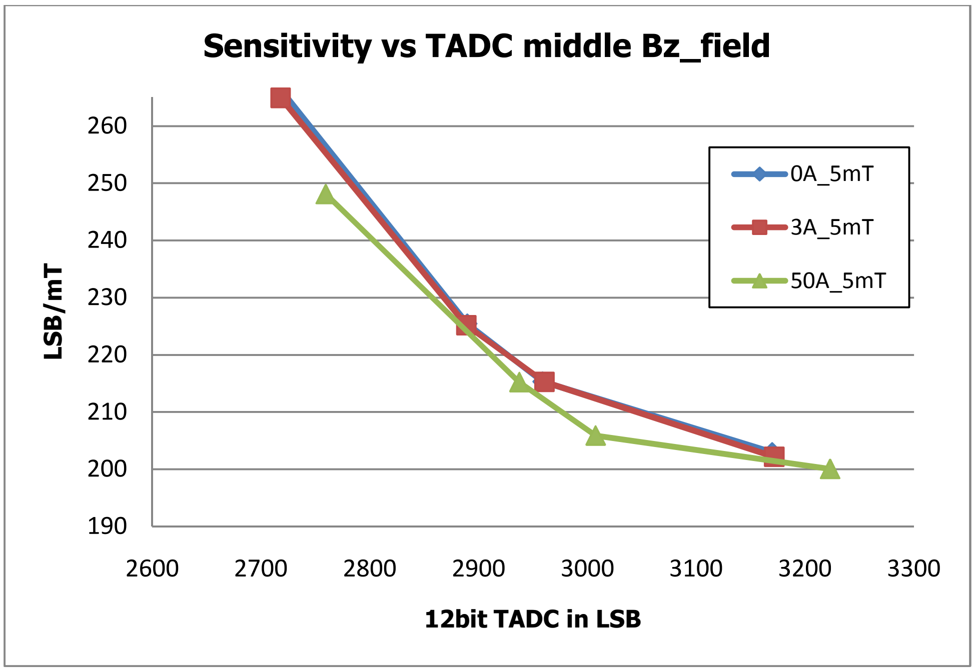


Figure 11.45 : Different sensitivity depending on the load current at the middle channel

Dependent on the load current there are different sensitivities at each channel. In this sample, mainly the middle channel shows a high dependency of the load current which takes effect to the sensitivity. Figure 11.44 shows the differences in sensitivity in LSB/mT at e varied load currents. Therefore the polynomial coefficients, which are describing the sensitivity functions, have been determined by the root mean square method. Following this, the sensitivity polynomials for the different load currents have to be calculated for the whole temperature range. The difference between the diverse sensitivities is shown in the Figure 11.46 to Figure 11.48.

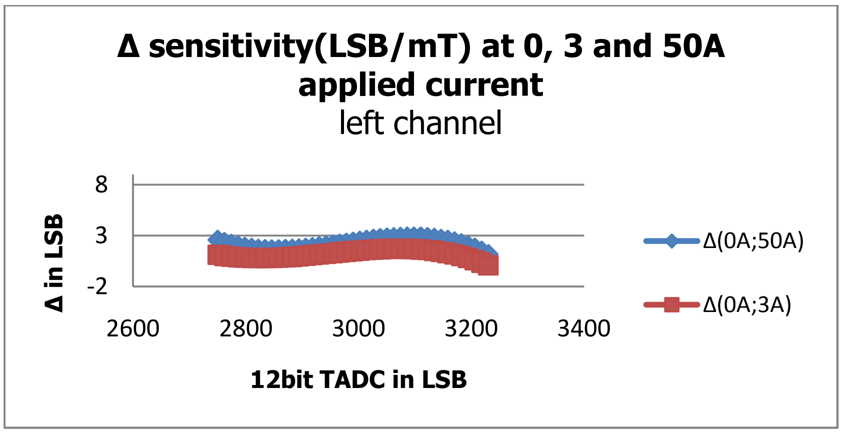


Figure 11.46 : Difference between Hall sensitivity with 3 and 50 amps applied at left

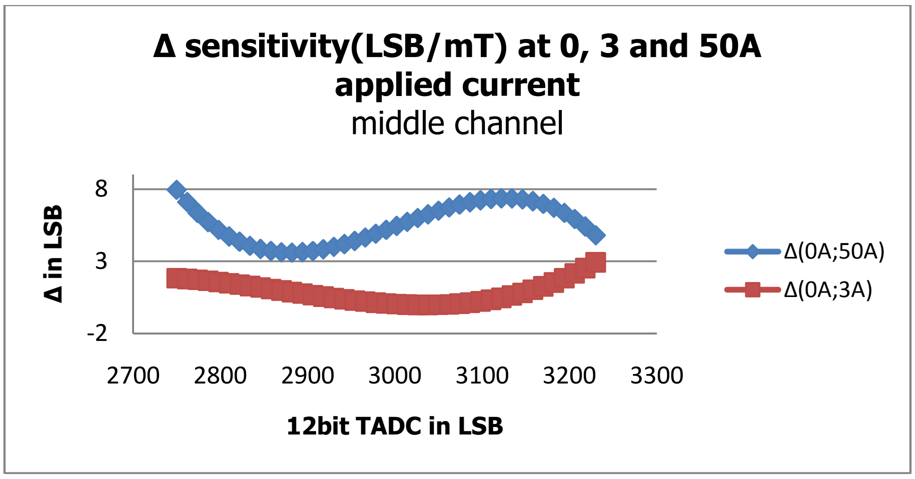


Figure 11.47 : Difference between Hall sensitivity with 3 and 50 amps applied at middle

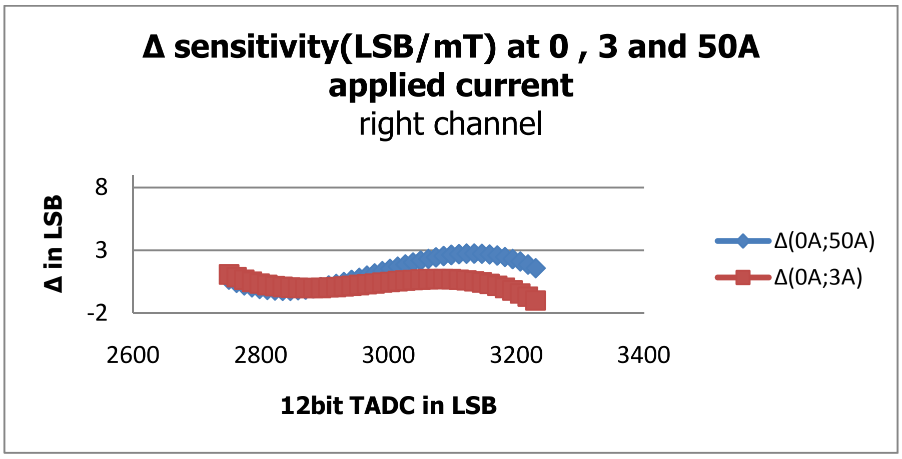


Figure 11.48 : Difference between Hall sensitivity with 3 and 50 amps applied at right

Figure 11.49 shows the error from the sensitivity in percent depending on how the field has been applied to the sensor. Equation 11.36 too Equation 11.38 describe how the sensitivities have been compared.

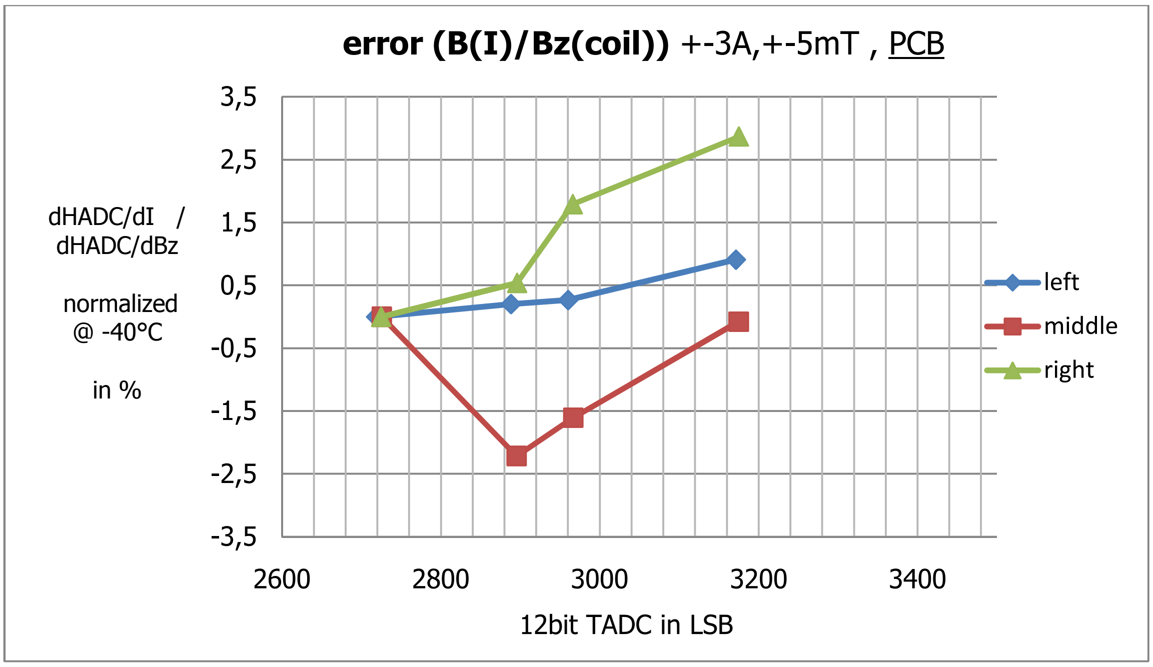


Figure 11.49 : Sensitivity error in percent depending on applied load current

As a first step the Hall sensitivity LSB/A has to be determined for the three measurement periods and normalized at room temperature.

$$S_{Bz(A)} = \left.\frac{dHADC}{dI}\right|_{normalized@RT}$$

Equation 11.36

Afterwards the HALL sensitivity dependent on the magnetic field, which is produced by the Helmholtz coil, has to be calculated and normalized at the same temperature as the sensitivity dependent on the load current.

$$S_{Bz(coil)} = \left.\frac{dHADC}{dB_z}\right|_{normalized@RT}$$

Equation 11.37

The error has been calculated by dividing the two normalized sensitivity vectors.

$$error = \left(\frac{S_{Bz(A)}}{S_{Bz(coil)}} \right)$$

Equation 11.38

Heating effects could be excluded because the temperature states respectively the TADC values are not distinguishable in these sensitivity measurements. Figure 11.50 shows the TADC values of each channel during the measurement. There is no difference to the TADC during superposing of the magnetic field.

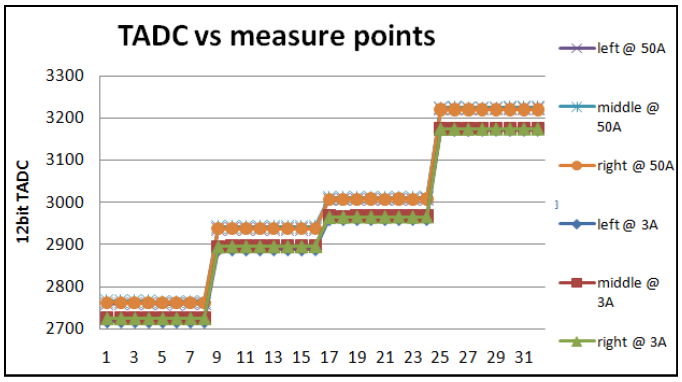


Figure 11.50 : TADC during the sensitivity - measuring

An effect which is only based on a stress effect could also be excluded because this effect has been canceled by comparing the sensitivities. The measurement results are given the evidence that the mismatch depends on a mechanical shift. In the test chip assembly process there can be some movements on the test board, where the silicon is mounted. As described in the hardware chapter the chip is placed on PCB and has been fixed with adhesive. A possible explication for the sensitivity problem could be a mechanical shift due to the relative high self-heating effect caused by the current. If the applied current produces hot spots at the copper rail, which is in between the PCB and the test chip, the connection between PCB and test chip may change. It is also possible that the position of the Hall plates may change due to mechanical drifts caused by heating. Also a slight rotation in the z direction of the sensor could happen and the magnetic field will not impact the chip in a perpendicular way anymore. This effect, respectively this error has been improved to a factor of 10 by changing the assembling of the sensor. Figure 11.51 shows the error between the Hall sensitivity in LSB/mT and LSB/A at the power copper test chip.

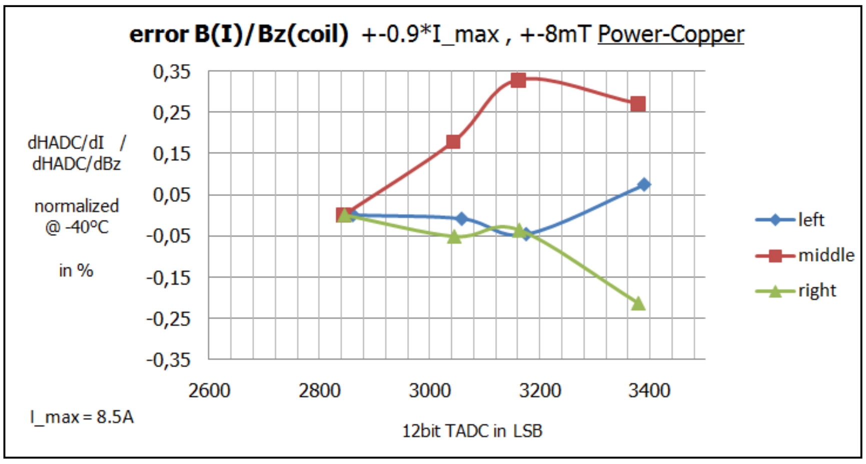


Figure 11.51 : Improved non linear effect by power copper sensor

The assembling distinguishes to the old PCB setup in a way that the current rail will be placed directly on the sensor. Therefore no glue is between the current rail and the chip anymore. The improvement of the mismatch with this assembling conforms the mechanical shift problem due to the adhesive.

12 Results and Outlook

Based on the measurements and the chip analyzes, the following forecast of how the Hall sensitivity and the offset can be compensated will be given. Further measurements and noise analysis has to be done for the redesigned product. Dependent on the sensor measurements, which has been analyzed in this thesis, a compensation regard can be given.

As described in the bias and operating setting chapter, the ideal bias settings for the Hall offset and sensitivity has been found. Also the temperature coefficient of the Hall plate has been decided as well as the settings for the analog stress sensor. Further analyses have to be done in future to find the ideal settings for the feedback loop of the continuous time sigma delta ADC by analyzing the spectrum of the Hall data. Another important topic which has to be analyzed is the behavior of the stress sensor from the redesigned current sensor. The stress effect is pre-compensated at the current sensor. The residual error has to be compensated at the output of the CLMCS. Therefore further stress analyzes will be necessary to allow compensating the sensitivity as well as the offset in respect to the stress signal from the ADC. For temperature compensation the following recommendation can be given.

The implemented digital chopping demodulation has shown a positive influence in respect to the offset. As described in the bias and operating setting chapter, the digital chopping demodulation can be enabled by setting the interface bit ‘Rvertical’. Therefore the signal at the output from the sensor is inverted. By calculating the difference of the inverted and non inverted Hall signal the offset decreases.

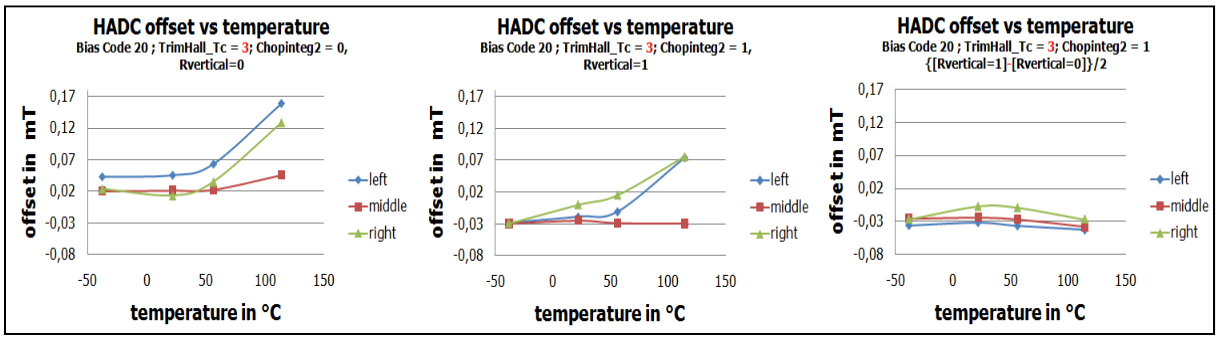


Figure 12.1 : Uncompensated Hall offset versus temperature

Figure 12.1 indicates the maximum range for the uncompensated offset in mT for the left channel. The left chart in Figure 12.1 represents the offset if the ‘Rvertical’ bit is not set to one. The middle chart shows the inverted output signal. The offset in the right chart represents the difference signal from the two other charts. An improvement of the offset by a factor of 10 can be shown if the different signal of the inverted and non inverted offset has been calculated. Figure 12.1 shows the amount of the offset if no compensation has been done.

Table 12.1 shows the improvement of the offset if the difference signal is used for the offset calculation in mT and percent related to the nominal current which corresponds to a magnetic field of 25mT.

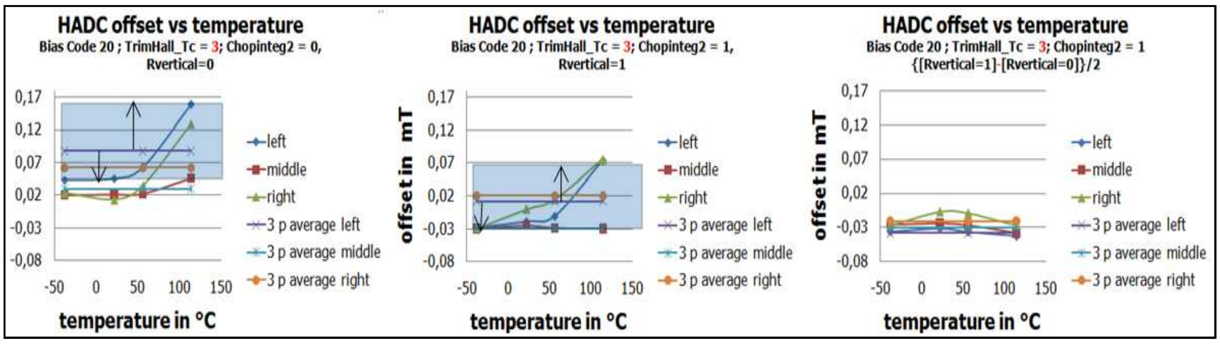


Figure 12.2 : Hall offset 'average-compensation'

Figure 12.2 shows reasonable compensation options for the offset. A linear offset compensation does not lead to a useful result because the second order part of the offset function is too dominant. Thus either the mean value over 3 temperature measure points has to be calculated and constantly subtracted from the Hall signal or a polynomial fit second order has to be done to calculate the offset in respect to the TADC values. Figure 12.2 indicates the offset compensation by calculating the average of three measure points. The principal is represented for the left channel.

Further measurements have to be done to find out if the quadratic coefficient of the offset function shows a systematic or statistical behavior. If a systematic third order coefficient can be determined, only three temperature measure points are necessary to calibrate the sensor with a adequate third order function.

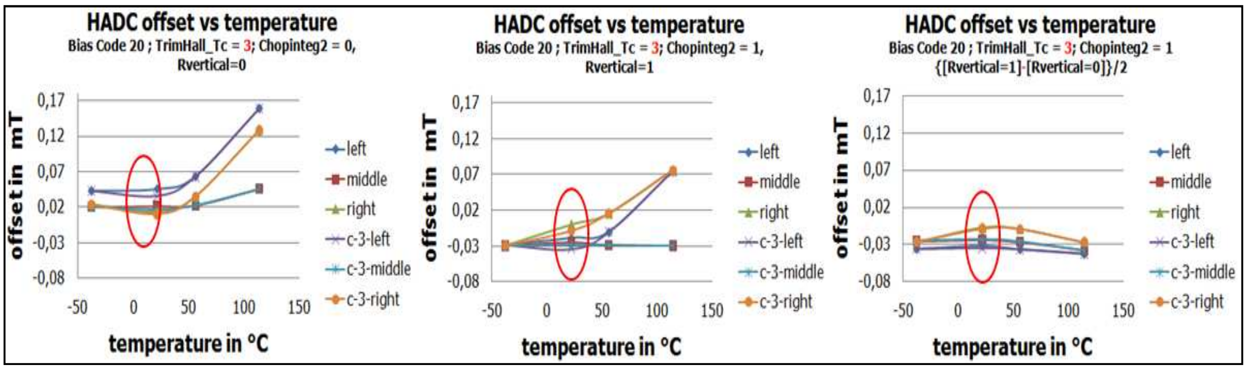


Figure 12.3 : Second order compensation of the Hall offset

Figure 12.3 shows the offset compensation with a quadratic function. Therefore three measure points are necessary to interpolate a compensation function.

Table 12.1 includes the maximum offset errors of the sensor dependent on the used compensation. The 'i-factor' represents the factor of improvement between the different compensations results.

Offset compensation

		Rvertival'-bit = 0			i-factor			Rvertival'-bit = 1			i-factor			(['Rvertival'-bit = 0] - ['Rvertival'-bit = 1])/2		
		left	middle	right	left	middle	right	left	midle	right	left	middle	right	left	middle	right
	offset in µT	**158,87**	**46,03**	**128,18**				**74,11**	**29,79**	**75,46**				**42,26**	**38,07**	**26,36**
no compensattion	offset in % related to nominal current(25mT)	0,635	0,184	0,513	2,1	1,5	1,7	0,296	0,119	0,302	1,8	1,3	2,9	0,169	0,152	0,105
	i-factor	2,2	2,8	1,9				1,2	5,9	1,4				6,4	4,6	2,0
compensation average of 3 temperatures	offset in µT	**70,71**	**16,26**	**65,88**				**62,39**	**5,06**	**55,00**				**6,64**	**8,21**	**13,24**
	offset in % related to nominal current(25mT)	0,011	0,003	0,011	1,1	3,2	1,2	0,010	0,001	0,009	0,4	40,6	0,2	0,027	0,033	0,053
	i-factor	7,4	3,4	22,0				4,1	1,1	6,5				2,1	51,7	7,9
	offset in µT	**9,54**	**4,84**	**2,99**				**15,24**	**4,43**	**8,46**				**3,24**	**-0,16**	**1,67**
2nd order compensation	offset in % related to nominal current(25mT)	0,038	0,019	0,012	1,6	1,1	2,8	0,061	0,018	0,034	4,7	27,9	5,1	0,013	0,0006	0,007

Table 12.1 : Hall Offset compensation

Figure 12.4 shows the Hall sensitivity function dependent on the temperature.

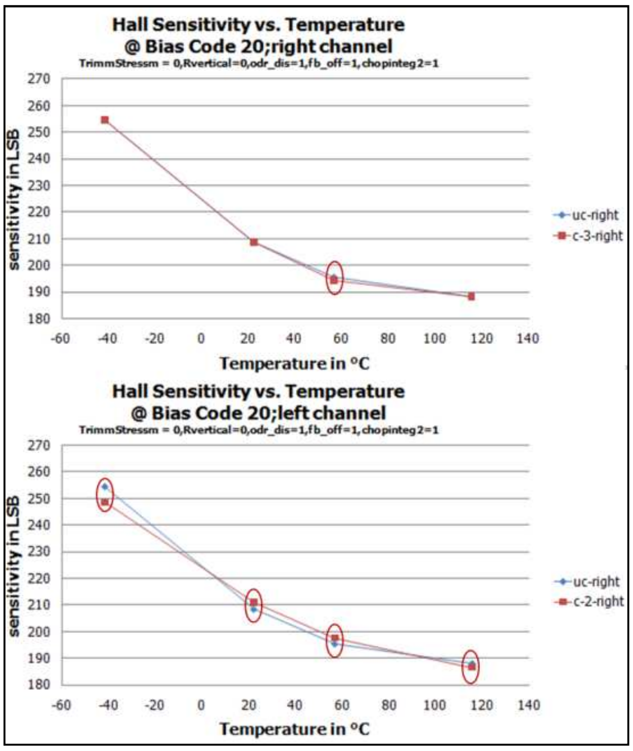


Figure 12.4 : Temperature Compensation of the Hall sensitivity

The sensitivity function describes a polynomial second order. Either a polynomial function second order has to be found to compensate the sensitivity or the systematically quadratic coefficient of the sensitivity can be determined. The different compensations are shown in Figure 12.4. If a systematical quadratic coefficient will be established, the calibration can be done at two temperatures. The determined fix second order part could be superposed to the linear function to compensate the sensitivity. Table 12.2 shows the error in percent, dependent on the compensation algorithm. The 'i-factor' row represents the ratio between the two different compensation modes.

Sensitivity compensation

		left	midddle	right
2nd order compensation 2 T-point fit (fix quadratic coeff.)	max error in %	2,73	2,81	2,41
	i-factor	4,7	5,3	4,2
2nd order compensation 3 T-point fit	max error in %	0,58	0,53	0,57

Table 12.2 : Hall sensitivity compensation

Further analyzes in respect of the stress sensor has to be done to allow a better compensation. Also multiplicities of offset and sensitivity measurements are necessary to get a statistic about the behavior of the Hall offset and the Hall sensitivity dependent on the temperature and stress.

13 Statutory Declaration

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am 02.02.2011 ....

[illegible]

(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

02.02.2011 ....

[illegible]

date (signature)

14 Acronyms and Abbreviations

UART	Universal Asynchronous Receiver Transmitter
FPGA	Field Programmable Gate Array
CIC	Cascaded Integrator Comb
IIR	Infinite Impulse response
FIR	Finite Impulse Response
ASIC	Application Specific Integrated Circuit
CLMCS	Core Less Magnetic Sensor
USB	Universal Serial Bus
RTL	Register Transfer Language
VHDL	Very High Description Language
SDRAM	Synchronous Dynamic random Access Memory
MM	Micro Module
HH	Helmholtz Coil
IFX	Infineon
SPI	Serial Peripheral Interface
ADC	Analog to Digital Converter
HADC	Analog to Digital Converter of the Hall signal
TADC	Analog to Digital Converter of the temperature signal
SADC	Analog to Digital Converter the mechanical stress signal
COMP	Comparator
DAC	Digital to Analog Converter
SYNC	Synchronization
DC	Direct current
AC	Alternating current
FS	full-scale
B2B	Board to Board Connector
MC	Modulation Circuit
JTAG	Join Test Action Group
PCB	Printed Circuit Board
GPIB	General Purpose Interface Bus

15 List of References

[1] Reinhart Weber, "Physik Teil I : Klassische Physik Experimentielle und theoretische Grundlagen", Teubner Verlag 2007

[2] A.V. Oppenheim , R.W Schafer, "Zeitdiskrete Signalverarbeitung", R.Oldenbourg Verlag München Wien 1999

[3] Elmar Schrüfer, "Elektrische Messtechnik", Hanser 2004

[4] Horst Kuchling, "Taschenbuch der Physik", Carl Hanser Verlag 2004

[5] Vorlesungsskript, "Theorie der Elektrotechnik", Institut für Grundlagen und Theorie der Elektrotechnik der TU-Graz, WS 2005/2006

[6] Martin Meyer, "Signalverarbeitubg", Vieweg+Teubner 2009

[7] Datasheet, "Synchronous DRAM MT48LC2M32B2-512Kx32x4 banks" Micron Technonogy 2001

[8] Datasheet, "DS099", Xilinx 2003 - 2004

[9] Mario Motz, Udo Ausserlechner, Wolfgang Scherr, Bernhar Schaffer
"An Integrated Magnetic Sensor with Two Continuous-Time ΔΣ-Converters and Stress Compensation Capability", IEEE International Solid-State Circuits Conference 2006

[10] Eugen B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", IEEE TRANSACTION ON ACCOUSTICS; SPEECH; AND SIGNAL PROCESSING, VOL. ASSP-29, NO. 2 APRIL 1981

[11] Infineon Technologies AG, "Linera Hall IC Current Sensor", Development Specification 2009

[12] ALTERA, "Understanding CIC Compensation Filters", April 2007

[13] Matthew P. Donadio, "CIC Filter Introduction", July 2000

[14] Horst Clausert, Gunther Wiesemann, Volker Hinrichsen, Jürgen Stenzel, "Grundgebiete der Elektrotechnik 1", Oldenbourg 2008

[15] Hans-Jochen Bartsch, "Tachenbuch Mathematischer Formeln", Fachbuchverlag Leipzig 2004

[17] Datasheet, "TLE4998", Infineon Technologies 2008

[18] Datasheet, USB 2.0 PHY IC "SMSC GT3200 Rev1.2b (05-07-03)", SMSC 2004

[19] Circuit Diagram, "SDRAM-Micromodule", trenc electronic 2005

[20] G. A. V. SOWTER, "Soft Magnetic Materials for Audio Transformers:History, Production and Applications*" , October 1987

[21] Richard Lyons, "Understanding cascaded integrator-comb filters", 3/31/2005 http://www.embedded.com/showArticle.jhtml?articleID=160400592

[22] Arbeitsunterlagen zur Vorlesung "Physik für Hörer der Studienrichtung Elektrotechnik und Hörer der Studienrichtung Telematik ", Institut für Experimentalphysik der Technischen Universität Graz Auflage WS 2004/2005

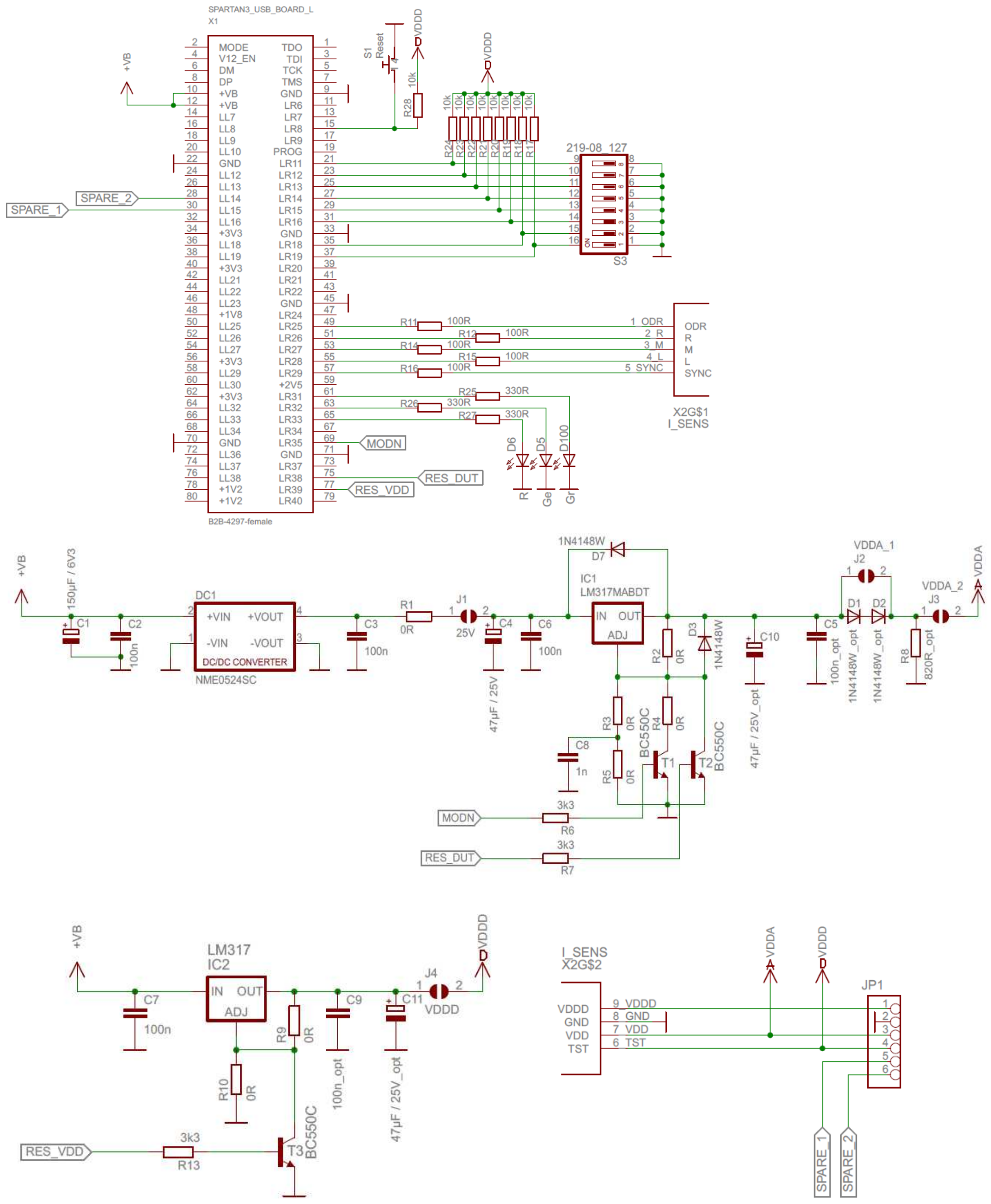
[23] Joris van Rantwijik, "Joris' WebPage", http://www.xs4all.nl/~rjoris/

16 List of Tables

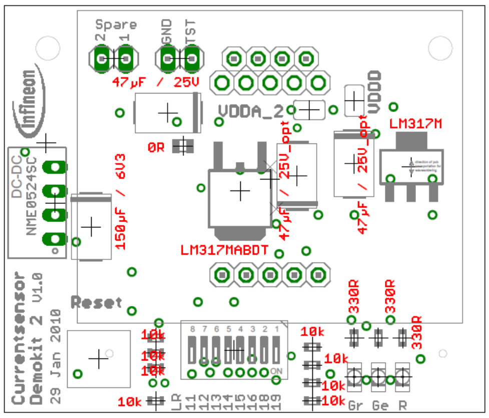
17 List of Figures

18 Appendix

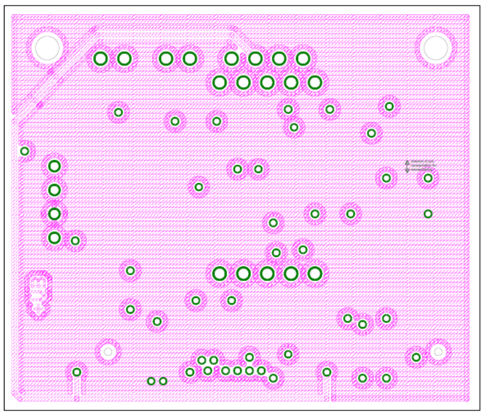
a) Modulation-board-schematic for the demonstrator tool



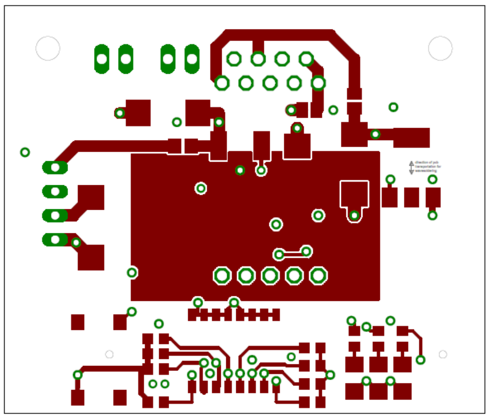
b) Modulation-board-layout fort the demonstrator tool



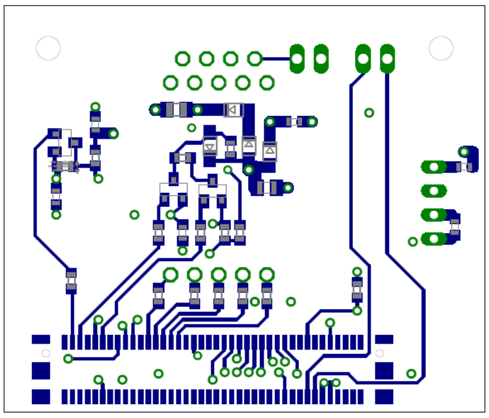
top



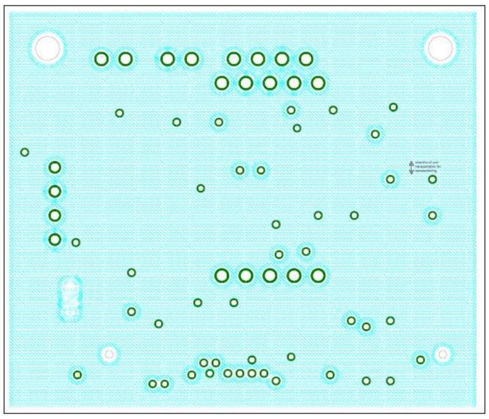
VDD



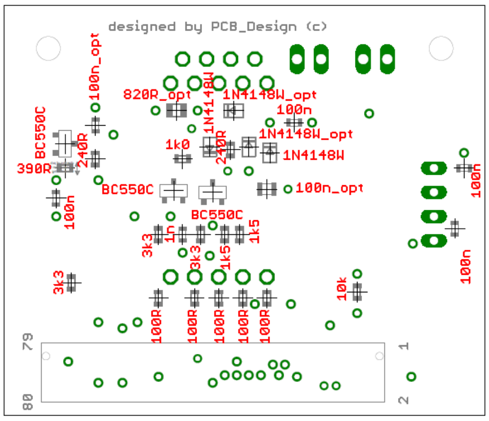
top



bottom

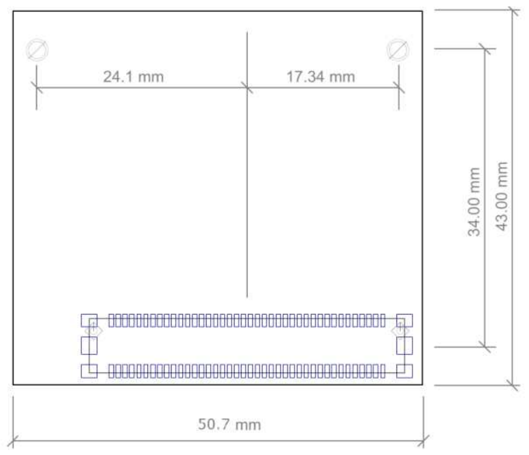


GND

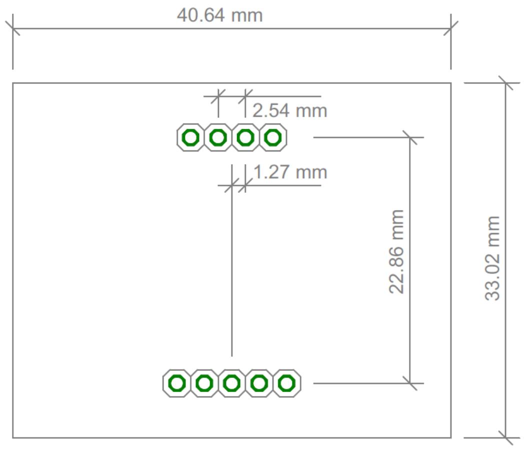


bottom

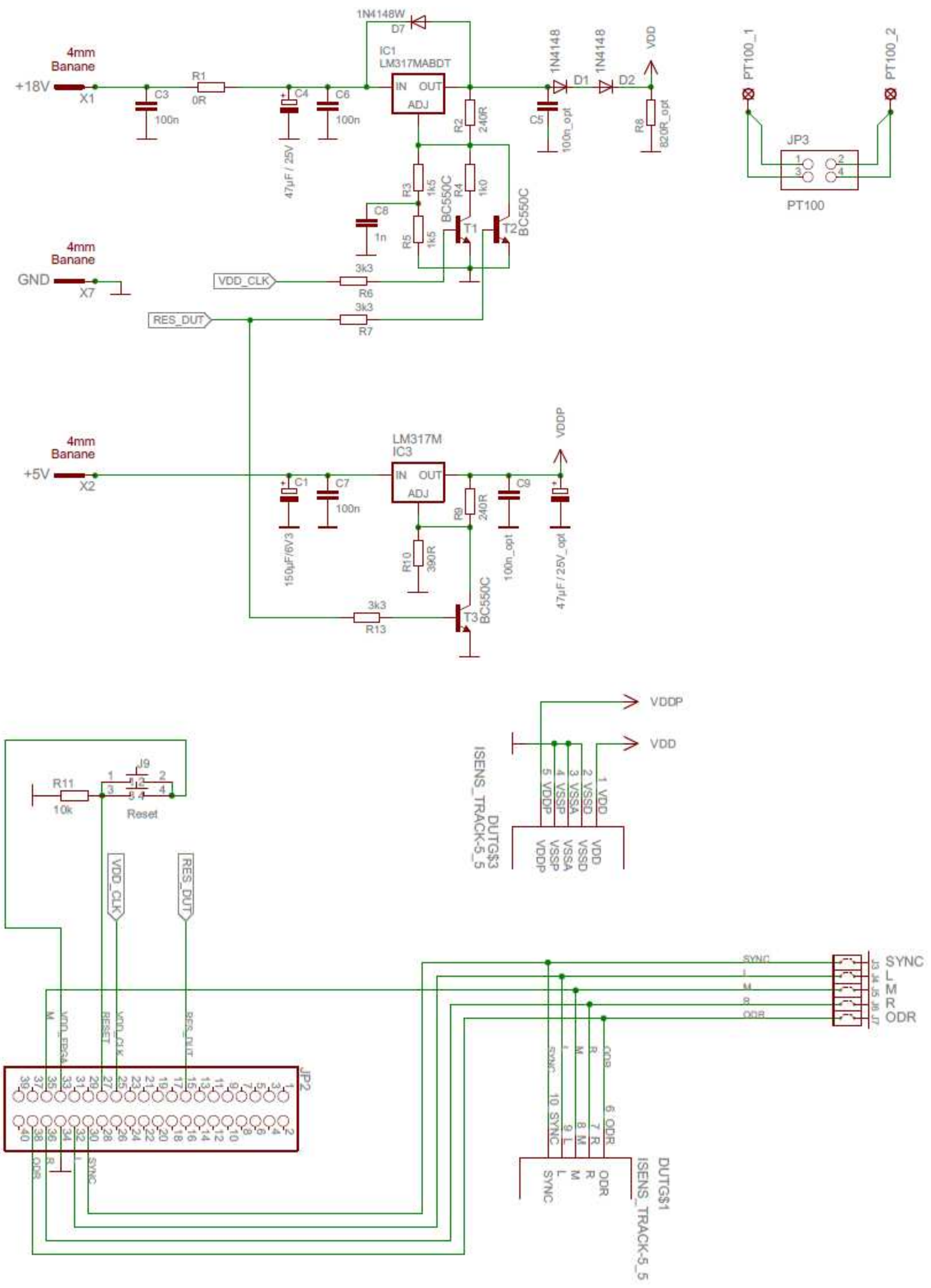
c) Modulation-board dimension



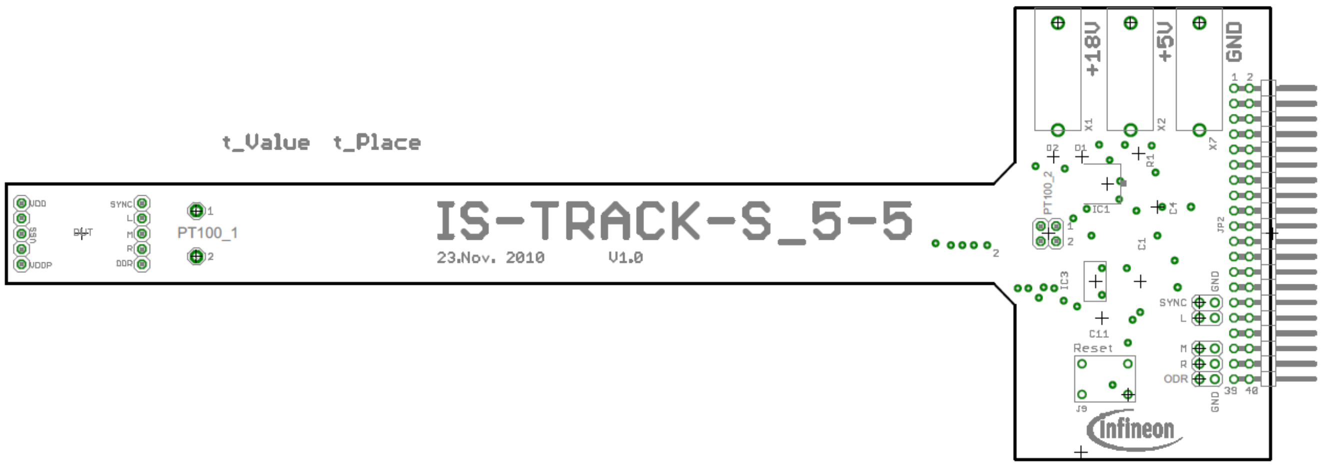
d) CLMCS – PCB Vi_1 dimension

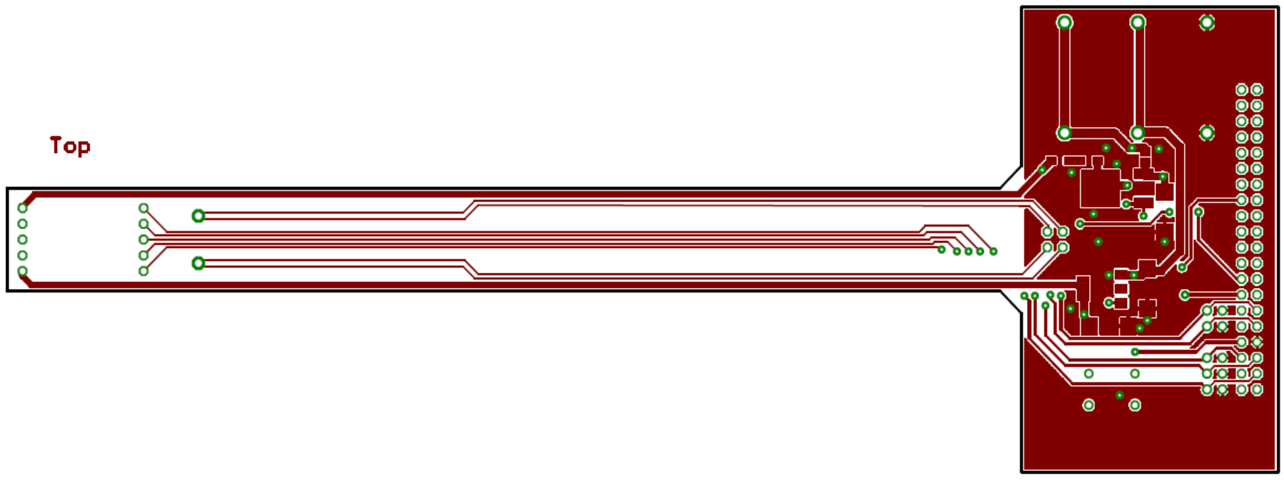


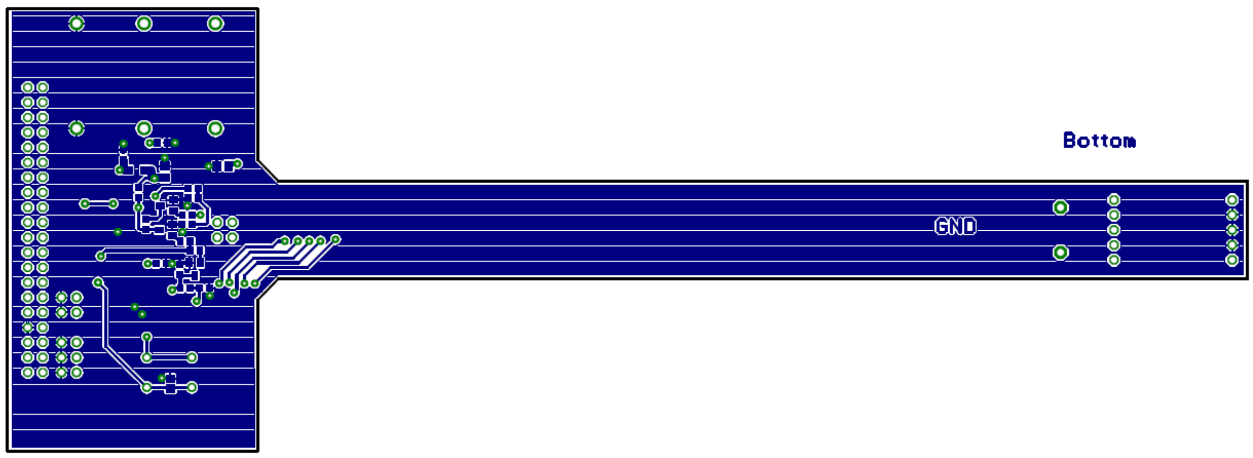
e) Measurement-hardware-schematic



f) Measurement-hardware-layout







g) List of material for the demonstration tool as well as the measurement hardware

QTY	Part	Value	Package	Description	Order #
1	C1	150µF / 6V3	D/7343-31R	Polarized Cap .3V,AVX - NOSD157M006R0070	1135182
5	C2,C3,C6,C7 C9	100nF	C0603K	Cap AVX - 06033C104JAT2A	1740614
3	C4,C10,C11	47µF / 25V	D/7343-31R	Cap KEMET - T491D476K025AT	1457517
1	C5	100nF	C0805K	100NF,AVX - 08053C104KAT2A	1740665
1	C8	1nF	C0603K	1000PF, 100V,AVX - 06031C102K4Z2A	1301714
4	D1,D2, D3,D4	1N4148W	SOD323-W	SOD-323,VISHAY SEMICONDUCTOR - BAS170WS/D5 Diode Schottky	1612320
2	IC1, IC3	LM317AEMP/NOP B	SMD	V REG, LINEAR, ADJ, SMD,NATIONAL SEMICONDUCTOR	1469084
2	T1,T2,T3	BC850C	SMD	TRANSISTOR SMD KLEINSIGNAL	1081241
2	R2, R9	240R	R0603	Resistor	
2	R3, R5	1k5	R0603	Resistor	
1	R4	1k0	R0603	Resistor	
3	R6,R7,R13	3k3	R0603	Resistor	
1	R8	820R	R0805	Resistor	
1	R10	390R	R0603	Resistor	
5	R11, R12, R14, R15, R16	100R	R0603	Resistor	
9	R17,R18,R19,R20,R21,R22,R23,R24,R28	10k	R0603	Resistor	
3	R25,R26,R27	330R	R0603	Resistor	
List of material for measurement hardware					
1	Connector Type:Board to Board	connector	Low profile	SAMTEC - SL-132-T-12	1668090
1	Connector Type:Board to Board	connector	Low profile	SAMTEC - BBL-132-G-E - HEADER, 2.54MM male	1667453

QTY	Part	Value	Package	Description	Order #
1	Connector Type:Board to Board	connector		FISCHER ELEKTRONIK - BL6.72Z - BUCHSENLEISTE,2X 36POL., 2,54MM	9728929
1	D10	Green	1206	LED	
1	D11	Yellow	1206	LED	
1	D12	red	1206	LED	
1	DC1	NME0524SC	NME	CONVERTER, DC/DC, SIL, 1W, 24V	1610431
1	S1 taster	Reset	B3S		

h) Netlist

```
# I/O constraints for XC3S1000 on TE0146 Micromodule
#NET "ram_io<0>"          LOC = "H15" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<1>"          LOC = "H14" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<2>"          LOC = "G16" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<3>"          LOC = "G14" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<4>"          LOC = "F14" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<5>"          LOC = "E15" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<6>"          LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "ram_io<7>"          LOC = "E13" | IOSTANDARD = LVCMOS33 | DRIVE = 24  | SLEW = FAST ;
#NET "diptaster_o<0>"     LOC = "G2" | IOSTANDARD = LVCMOS33;#lr19
#NET "diptaster_o<1>"     LOC = "G1" | IOSTANDARD = LVCMOS33;#lr18
#NET "diptaster_o<2>"     LOC = "G3" | IOSTANDARD = LVCMOS33;#lr16
#NET "diptaster_o<3>"     LOC = "G4" | IOSTANDARD = LVCMOS33;#lr15
#NET "diptaster_o<4>"     LOC = "F2" | IOSTANDARD = LVCMOS33;#lr14
#NET "diptaster_o<5>"     LOC = "F3" | IOSTANDARD = LVCMOS33;#lr13
#NET "diptaster_o<6>"     LOC = "E1" | IOSTANDARD = LVCMOS33;#lr12
#NET "diptaster_o<7>"     LOC = "E2" | IOSTANDARD = LVCMOS33;#lr11
NET "phy_clkout"          LOC = "D9" | IOSTANDARD = LVCMOS33 ;
NET "phy_databus16_8"     LOC = "B14" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_datain<0>"       LOC = "A8" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<1>"       LOC = "C8" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<2>"       LOC = "C7" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<3>"       LOC = "A7" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<4>"       LOC = "B7" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<5>"       LOC = "B5" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<6>"       LOC = "C5" | IOSTANDARD = LVCMOS33 ;
NET "phy_datain<7>"       LOC = "A4" | IOSTANDARD = LVCMOS33 ;
NET "phy_dataout<0>"      LOC = "A13" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<1>"      LOC = "B11" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<2>"      LOC = "C10" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<3>"      LOC = "B10" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<4>"       LOC = "A10" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<5>"       LOC = "C9" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<6>"       LOC = "A9" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_dataout<7>"       LOC = "B8" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_linestate<0>"     LOC = "A3" | IOSTANDARD = LVCMOS33 ;
NET "phy_linestate<1>"     LOC = "A5" | IOSTANDARD = LVCMOS33 ;
NET "phy_opmode<1>"        LOC = "C6" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_opmode<0>"        LOC = "T10" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_reset"            LOC = "B4" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_rxactive"         LOC = "C11" | IOSTANDARD = LVCMOS33 ;
NET "phy_rxerror"          LOC = "C12" | IOSTANDARD = LVCMOS33 ;
NET "phy_rxvalid"          LOC = "A12" | IOSTANDARD = LVCMOS33 ;
NET "phy_termselect"       LOC = "B6" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_txready"          LOC = "A14" | IOSTANDARD = LVCMOS33 ;
NET "phy_txvalid"          LOC = "B12" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "phy_xcvrselect"       LOC = "D6" | IOSTANDARD = LVCMOS33 | DRIVE = 12 | SLEW = FAST ;
NET "led"                  LOC = "N6" | IOSTANDARD = LVCMOS25 | DRIVE = 12;

NET "phy_clkout" TNM_NET = "phy_clkout";
TIMESPEC "TS_clk" =        PERIOD "phy_clkout" 16 ns HIGH 50 %;
#TIMESPEC "TS_clk" =       PERIOD "phy_clkout" 14 ns HIGH 50 %;

NET "phy_datain<0>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<1>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<2>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<3>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<4>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<5>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<6>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_datain<7>"        OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_dataout<0>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<1>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<2>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<3>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<4>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<5>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<6>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_dataout<7>"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_linestate<0>"     OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_linestate<1>"     OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_opmode<1>"        OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_opmode<0>"        OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_reset"            OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_rxactive"         OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_rxerror"          OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_rxvalid"          OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_termselect"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_txready"          OFFSET = IN 6 ns AFTER "phy_clkout" ;
NET "phy_txvalid"          OFFSET = OUT 6 ns BEFORE "phy_clkout" ;
NET "phy_xcvrselect"       OFFSET = OUT 6 ns BEFORE "phy_clkout" ;

NET "rst_n_i"              LOC = "D3" | IOSTANDARD = LVCMOS33; #lr8   F4 for measurement hardware
NET "odr_i"                LOC = "J2"  | IOSTANDARD = LVCMOS33; #lr25 G2 for measurement hardware
NET "out_l_i"              LOC = "L3" | IOSTANDARD = LVCMOS33; #lr28 G3 for measurement hardware
NET "out_m_i"              LOC = "K1" | IOSTANDARD = LVCMOS33; #lr27 J4  for measurement hardware
NET "out_r_i"              LOC = "K2" | IOSTANDARD = LVCMOS33; #lr26 G1 for measurement hardware
NET "vdd_clk_o"            LOC = "L5" | IOSTANDARD = LVCMOS33; #lr35 E4 for measurement hardware
NET "sync_i"               LOC = "L2" | IOSTANDARD = LVCMOS33; #lr29 G4 for measurement hardware
NET "supplyoffdut_o"       LOC = "N2" | IOSTANDARD = LVCMOS33; #lr38 D7 for measurement hardware
NET "supplyoff_o"          LOC = "N3" | IOSTANDARD = LVCMOS33; #lr39 D7 for measurement hardware

NET "test1_o"              LOC = "M1" | IOSTANDARD = LVCMOS33; #led red        lr33
NET "test2_o"              LOC = "M2" | IOSTANDARD = LVCMOS33; #led yellow     lr32
NET "test3_o"              LOC = "M3" | IOSTANDARD = LVCMOS33; #led green      lr31
NET "myclk2_o"             LOC = "E1" | IOSTANDARD = LVCMOS33; #LR12 - phyclkout
###sdram
NET "clk_o"                LOC = "H16" | IOSTANDARD = LVCMOS33;
NET "cke_o"                LOC = "J16" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<0>"         LOC = "E11" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<1>"         LOC = "D11" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<2>"         LOC = "D12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<3>"         LOC = "G12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<4>"         LOC = "D15" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<5>"         LOC = "F12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<6>"         LOC = "F14" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<7>"         LOC = "F15" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<8>"         LOC = "G16" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<9>"         LOC = "E15" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<10>"        LOC = "E16" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<11>"        LOC = "E14" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<12>"        LOC = "E13" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<13>"        LOC = "D16" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<14>"        LOC = "D14" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<15>"        LOC = "C15" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<16>"        LOC = "L12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<17>"        LOC = "L14" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<18>"        LOC = "M13" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<19>"        LOC = "P15" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<20>"        LOC = "N11" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<21>"        LOC = "P10" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<22>"        LOC = "R10" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<23>"        LOC = "T9" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<24>"        LOC = "R11" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<25>"        LOC = "P11" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<26>"         LOC = "T12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<27>"         LOC = "R12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<28>"         LOC = "P12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<29>"         LOC = "N12" | IOSTANDARD = LVCMOS33;
NET "dq31_0_io<30>"         LOC = "T14" | IOSTANDARD = LVCMOS33;
NET "dqm3_0_o<0>"           LOC = "F13" | IOSTANDARD = LVCMOS33;
NET "dqm3_0_o<1>"           LOC = "G15"| IOSTANDARD = LVCMOS33;
NET "dqm3_0_o<2>"           LOC = "L15" | IOSTANDARD = LVCMOS33;
NET "dqm3_0_o<3>"           LOC = "N14"| IOSTANDARD = LVCMOS33;
NET "ba01_o<0>"             LOC = "H14"| IOSTANDARD = LVCMOS33;
NET "ba01_o<1>"             LOC = "K13" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<0>"            LOC = "J14" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<1>"            LOC = "H15  | IOSTANDARD = LVCMOS33;
NET "a10_0_o<2>"            LOC = "L13" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<3>"            LOC = "N16" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<4>"            LOC = "N15" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<5>"            LOC = "M15" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<6>"            LOC = "M16" | IOSTANDARD = LVCMOS33;
NET "a10_0_o<7>"            LOC = "K14"  | IOSTANDARD = LVCMOS33;
NET "a10_0_o<8>"            LOC = "K16"  | IOSTANDARD = LVCMOS33;
NET "a10_0_o<9>"            LOC = "K15"  | IOSTANDARD = LVCMOS33;
NET "a10_0_o<10>"           LOC = "J13"  | IOSTANDARD = LVCMOS33;
NET "comm_o<3>"             LOC = "K12"  | IOSTANDARD = LVCMOS33; # cs
NET "comm_o<2>"             LOC = "G13"  | IOSTANDARD = LVCMOS33; # ras
NET "comm_o<1>"             LOC = "G14"  | IOSTANDARD = LVCMOS33; # cas
NET "comm_o<0>"             LOC = "H13"  | IOSTANDARD = LVCMOS33; # we
NET "sync_o"                LOC = "D8"  | IOSTANDARD = LVCMOS33;  # ll7 spare_2
NET "txrdy_o"               LOC = "D1"  | IOSTANDARD = LVCMOS33;  # ll7 spare_1

# debug
#NET "test4_o"              LOC = "D15" | IOSTANDARD = LVCMOS33; # RL7
#NET "test5_o"              LOC = "E14" | IOSTANDARD = LVCMOS33; # RL9
#NET "test6_o"              LOC = "E15" | IOSTANDARD = LVCMOS33; # RL11
#NET "test7_o"              LOC = "E16" | IOSTANDARD = LVCMOS33; # RL12
#NET "test8_o"              LOC = "F14" | IOSTANDARD = LVCMOS33; # RL13 8
#net  "myclk_o"             LOC = "F15" | IOSTANDARD = LVCMOS33; # RL14
#net "myclk2_o"             LOC = "P8" | IOSTANDARD = LVCMOS33;  # LL38
#NET "test5_o"              LOC = "G3" | IOSTANDARD = LVCMOS33; # LR16 - usb_txval
#NET "test6_o"              LOC = "G2" | IOSTANDARD = LVCMOS33; # LR19 - usbrxval
#NET "test7_o"              LOC = "F2" | IOSTANDARD = LVCMOS33; # LR14 - usbsend_o
```

i) Setup information file

(c) Hans Hübner
Code to generate the setup information file to install the software for the usb communication with the demonstrator tool for system based on windows. [23]

```
[Version]
ignature="$Windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%JORIS%
LayoutFile=layout.inf
DriverVer=10/15/1999,5.0.2153.1
[Manufacturer]
%JORIS%=JORIS
[JORIS]
%FPGASER%=Reader, USB\VID_FB9A&PID_FB9A
[Reader_Install.NTx86]
;Windows2000
[DestinationDirs]
DefaultDestDir=12
Reader.NT.Copy=12
[Reader.NT]
CopyFiles=Reader.NT.Copy
AddReg=Reader.NT.AddReg
[Reader.NT.Copy]
usbser.sys
[Reader.NT.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"
[Reader.NT.Services]
AddService = usbser, 0x00000002, Service_Inst
[Service_Inst]
DisplayName = %Serial.SvcDesc%
ServiceType = 1 ; SERVICE_KERNEL_DRIVER
StartType = 3 ; SERVICE_DEMAND_START
ErrorControl = 1 ; SERVICE_ERROR_NORMAL
ServiceBinary = %12%\usbser.sys
```

```
LoadOrderGroup = Base
[Strings]
JORIS = "Joris"
FPGASER = "FPGA Serial"
Serial.SvcDesc = "USB Serial emulation driver" [23]
```