MULTI PROJEKT CHIP GRUPPE

AUSGABE: 46

MPC

MULTI PROJEKT CHIP GRUPPE
BADEN-WÜRTTEMBERG

Herausgeber: Hochschule Ulm **Ausgabe:** 46 **ISSN** 1868-9221 **Workshop:** Furtwangen Juli 2011

Cooperating Organisation
Solid-State Circuit Society Chapter
IEEE German Section

Inhaltsverzeichnis

Tagungsband zum Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie.

Herausgeber: Gerhard Forster, Hochschule Ulm, Prittwitzstraße 10, D-89075 Ulm

Diesen Workshopband und alle bisherigen Bände finden Sie im Internet unter:

http://www.mpc.belwue.de

hochschule aschaffenburg
university of applied sciences

A Ring-Oscillator-Based Temperature Sensor with Scalable Resolution

Andreas Brönner, Daniel Fuchs, Ulrich Brunsmann

***Abstract*—A ring-oscillator-based digital temperature sensor with scalable resolution is presented. Through added temperature sensitive resistors the thermal sensitivity is shaped. The circuit was fabricated in AMS 0.35 µm CMOS technology. Unlike conventional temperature sensors relying on semiconductor pn-junctions and ADCs, the proposed sensor exploits the frequency difference of two ring oscillators as a way of converting temperature to a digital output. The experimental results show that the resolution can be scaled from 0.24 °C to 55 µ°C at the expense of sampling rate over a temperature range from -25 °C to 75 °C. We present the basics of operation, the design and layout of the chip and an evaluation of performance data.**

***Index Terms*—CMOS, digital, temperature sensor, RC load, scalable resolution, ring oscillator.**

I. INTRODUCTION

In recent years all digital temperature sensors became more interesting. The tendency towards decreasing feature size and associated reduction in transistor gate-oxide thickness forces the system voltage to decrease, making analog design difficult [1]. Analog smart temperature designs usually adopt the parasitic substrate or lateral bipolar transistor for temperature sensing taking advantage of bandgap reference [2]. Traditional analog-to-digital converters are utilized for converting a voltage or current into subsequent digital output code. In order to reduce measurement errors, additional calibration circuits at the expense of chip area and power consumption proved to be necessary [3].

Digital temperature sensors however are mostly based on temperature effects of the mobility of charge carriers and threshold voltage of the MOSFET in order to produce a pulse width variation proportional to the test temperature change. Besides TDC-based temperature sensors, generating a pulse width proportional to test temperature fed into a time-to-digital converter (TDC) for output coding [4], it is possible to use the sensitivity of the pulse-shrinking-TDC output code to measure temperature directly without any temperature-to-pulse generator [5]. Temperature dependent propagation delays of the rising and falling edges through different gates of the pulse-shrinking TDC cause comparative temperature dependent pulse shrinking, generating a change in the digital output code. Recently starved ring oscillators, implemented as temperature dependent elements for special applications like Radio Frequency Identification Tags [6] or U-Healthcare [7], feature low power consumption (30 nW) with medium resolution (0.18 °C/LSB). Usually temperature sensitive resistors steer current through ring oscillator cells. In [8], a frequency-to-digital converter based temperature sensor is presented. Temperature is measured using the difference in frequencies between a temperature sensitive oscillator and a temperature insensitive oscillator reaching a high sampling rate of 366 kS/s.

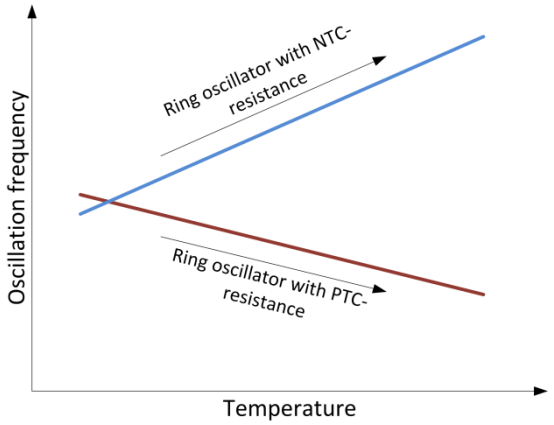


Fig. 1. Conceptual graph of temperature dependency of the oscillation frequency of ring oscillators with NTC- and PTC-resistance.

Smart temperature sensor performance requirements differ as much as application fields, for example, engine limit monitoring with low accuracy [9], temperature monitoring for home or office electronics, mobile phones, PDAs with medium accuracy [10] up to medical equipment requiring high accuracy [11].

Addressing the issue of wide range of applications in this paper a smart temperature sensor with scalable resolution is proposed. Fig. 1 shows an ideal temperature-frequency-relationship of two ring oscillators applied. The ring oscillator employing a PTC-

Andreas Brönner , andreas.broenner@h-ab.de, Daniel Fuchs, daniel.fuchs@h-ab.de, and Ulrich Brunsmann, ulrich.brunsmann@h-ab.de, are with University of applied sciences Aschaffenburg, engineering department, laboratory of electronic devices, Würzburger Straße 45, 63743 Aschaffenburg.

resistance increases propagation delay and thus decreases frequency as temperature increases. This oscillation frequency is used to generate a time slot within a counter counting the number of clock cycles received from the second ring oscillator. The width of this time slot can be coarsely scaled by a frequency divider. The resulting digital code is proportional to the test temperature.

The paper is organized as follows: in section II we use delay models to describe the operating principle of the proposed delay elements before we present how the resolution can be scaled. Performance data are shown in section III followed by the conclusion in section IV.

II. Proposed CMOS Temperature Sensor

In this section we describe the ring oscillator design modeling using RC delay models. We consider in particular, how the resolution can be scaled.

A. 2nd Order Delay Estimation

Since propagation delay is one of the most critical performance parameters in CMOS digital circuits, much effort had been devoted to the extraction of accurate, analytical expressions for timing models of basic circuits. Besides the RC delay model, the Elmore delay model gains importance [12].

RC delay models approximate the nonlinear transistor current-voltage and capacity-voltage characteristics with average resistance and capacitance over the switching range of the MOSFET gate. This approximation works remarkably well for delay estimation despite its obvious limitation in predicting detailed analog behavior.

Our ring oscillators consist of an odd number of inverter gates including a subsequent on chip RC circuit. The equivalent RC circuit of one stage is shown in Fig. 2, input capacitances of stage one are neglected as well as wire capacitance. Analyzing the propagation delay of the transition from high to low t_{PHL}, we summarize the capacitors C_e, C_{IN2P} and C_{IN2N} to C_{le} including the extra capacitor C_e. Output capacitors of the first stage are merged to C_l achieving a 2nd order RC circuit. Using the inverse Laplace transformation of the systems transfer function, the step response can be calculated.

$$V_{OUT}(t) = V_{DD}\frac{\tau_1 e^{-t/\tau_1} - \tau_2 e^{-t/\tau_2}}{\tau_1 - \tau_2}$$

τ_1 and τ_2 depend on the resistances and capacitances:

$$\tau_{1,2} = \frac{R_n C_l + (R_n + R_e)C_{le}}{2}\left(1 \pm \sqrt{1 - \frac{4\frac{R_e}{R_n}\frac{C_{le}}{C_l}}{\left[1 + \left(1 + \frac{R_e}{R_n}\right)\frac{C_{le}}{C_l}\right]^2}}\right)$$

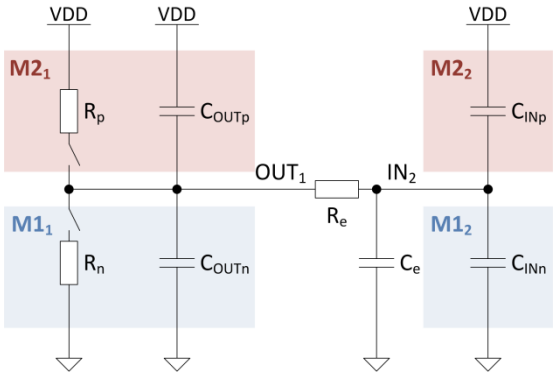


Fig. 2. Equivalent circuit for two inverters with extra RC network in between, designed as ring oscillator stage.

The system can be approximated as a 1st order system with a single time constant [12], i.e. the second order RC combination can be estimated as a first order system with one single time constant:

$$\tau = \tau_1 + \tau_2 = R_n C_l + (R_n + R_e)C_{le}\ .$$

If the two time constants are equal ($\tau_1 = \tau_2$) the error in estimated propagation delay reaches the worst case value of less than 15%. If one time constant is remarkably bigger than the other, error is less than 7 % [12]. It is possible to control the propagation delay of the stage if the extra resistance and capacitance are designed with respect to the time constant. Temperature dependant variation of resistance leads to chances of propagation delay and oscillation frequency according to temperature.

Initially, we look at temperature effects influencing the MOSFET current. If silicon is used as base material, the mobility of charge carriers μ is a function of temperature and the current through the switches varies accordingly [13]. Operating the MOSFETS above the threshold voltage V_T, which also varies with temperature, and above cryogenic temperatures, well known empirical formula apply for silicon [14]:

$$\mu(T) = \mu(300K)\left(\frac{T}{300K}\right)^{-a}$$

$$a = 1{,}5 \ldots 2{,}5$$

$$\frac{\Delta V_T}{\Delta T} = -1\frac{mV}{°C} \ldots -3\frac{mV}{°C}$$

These effects have been used to build a digital sensor and, as they inversely modify the switching current, also for thermal compensation, e.g. by [4, 5, 15]. The combined temperature effects result in a current with negative temperature coefficient and propagation delay with positive temperature coefficient. Ring oscillator frequency decreases as temperature increases.

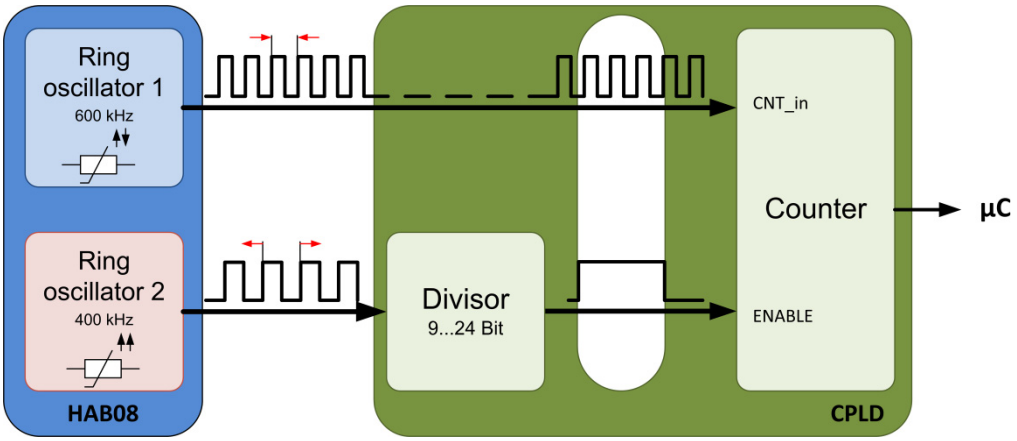


Fig. 3. Operating principle of our ring-oscillator-based digital sensor. The frequency-temperature-relationship is implemented according to Fig. 1.

To control the temperature coefficient of the ring oscillator frequency, we instead propose to use an extra resistance and capacitance which may be described by the BSIM3v3 equations [16]:

$$R(T) = R(TNOM)(1 + TCR1(T - TNOM))$$

$$C(T) = C(TNOM)(1 + TCC1(T - TNOM))$$

The temperature coefficients of different layers for resistor (TCR1) and capacitor (TCC1) given in the manufacturers design handbook are essential for the relation between frequency and temperature. In order to get the frequency characteristics shown in Fig. 1, dimensioning of the extra resistor is important. Lower MOSFET carrier mobility forces the frequency to decrease with rising temperature. This effect can be either reinforced with extra PTC resistance, or not only be compensated but excelled by implementing an extra NTC resistance in order to get a rising frequency curve with temperature. In our test chip, the negative temperature coefficient of frequency caused by MOSFET current is damped using a poly resistor R_e in one oscillator, while being strengthened using an n-well resistor R_e in the other.

B. Scaling the Resolution

The structure of our proposed temperature sensor is shown in Fig. 3. The HAB08 test chip containing two ring oscillators has already been described [5]. The oscillators differ in their fundamental frequency and temperature coefficient of frequency, achieved by scaling of the additional resistor and capacitor. In order to reach sub-degree resolution, an intelligent handling of ring oscillator signals is needed. Simple analysis of frequency difference between the two oscillators would stick the resolution to a predefined value, usually several degrees [8]. Hence, we use a different method to convert the frequency into a digital code proportional to the test temperature. Pulses of the ring oscillator frequency, having a lower or negative temperature coefficient, are fed into a configurable frequency divider, downsizing frequency by 2^n, where n is set by a microcontroller. The output signal generates a time slot whilst a counter counts the number of clock cycles received from the second ring oscillator with positive temperature coefficient. Increasing the time slot width and increasing the number of pulses to be counted at the same time are the main features of our sensor system, connecting the resulting digital output code proportional to the measured temperature. Temperature resolution of our sensor system can be scaled by changing the time slot width using the frequency divider. We can change the divisor from 2^9 to 2^{24} in order to scale the resolution and so the accuracy from 3.75 °C to 0.06 °C, considering $3\sigma_{max}$ maximum standard deviation, see Table 1. With longer time slot, our sampling rate is scaled from 204.8 Hz to 0.05 Hz. Higher resolution is achieved at the expense of the measurement rate, as shown in Fig. 8.

III. Measurement Data

The test chip was fabricated using an AMS 0.35 µm CMOS process. Beside the proposed ring oscillators the HAB08 test chip contains two delay lines, a counter and a pulse shrinking TDC. The chip was designed for advanced laboratory courses within our research projects on smart sensors [5]. The ring oscillator sensor occupies 0.21 mm² of chip area while complete chip size is 3 mm², see Fig. 5. In this section, we present the temperature sensor system including experimental setup and measurement results.

A. Experimental Setup

In order to evaluate the sensor system, a configurable printed evaluation board for the HAB08 test chip was designed to operate the ring oscillators as single systems or a complete sensor system. The step-down voltage regulator LM2671M-3.3 (National Semiconductor) with a constant output voltage of 3.3 V is used to simulate operating conditions and allows the board

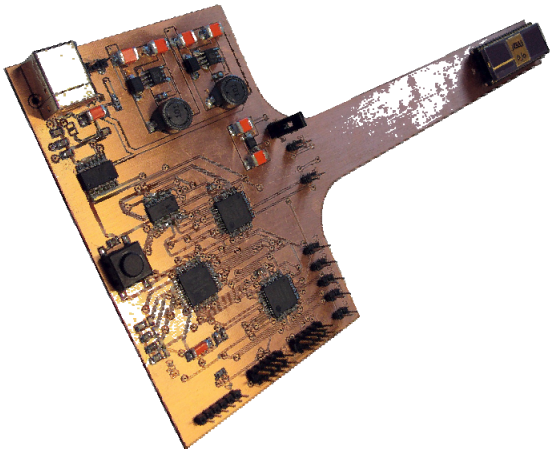


Fig. 4. Evaluation board for our temperature sensor system showing a test chip in the background.

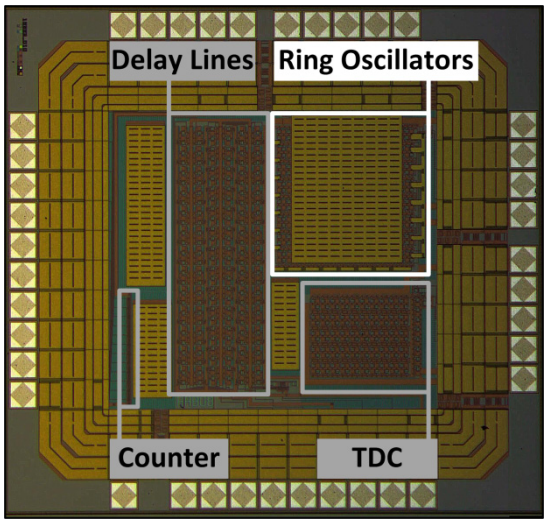


Fig. 5. Microphotograph of the test chip with highlighted structures.

to be battery powered. Two CPLDs (EPM3064, Altera) and a microcontroller (PIC18F45K20, Microchip) are being utilized. As shown in Fig. 3, one CPLD is configured to divide the ring oscillator frequency, while the second CPLD counts incoming pulses and feeds the digital output to the microcontroller, where the information is evaluated and sent to a PC via a RS232 to USB converter (FT232RL). The divider is controlled by changing the configuration of a 16:1 multiplexer between the Divider-CPLD and the Counter-CPLD. The current multiplexer position is connected to microcontroller port signals, shifted by keystroke.

The finger of the evaluation board containing the chip was fitted into a temperature dry well calibrator (CTD9300-165, WIKA). The latter was certified by a DAkkS[1] laboratory and ensures traceability of our temperature calibration to the german PTB[2] standard. In order to achieve a temperature stability of 0.02 °C and measurement errors of 0.1 °C as specified by the manufacturer of the calibrator, the sensor is placed in the lower third of the calibrator bushing. A reference PT100 sensor class AA is set on top of the HAB08 test chip package for the purpose of measuring the package temperature during calibration. The resistor value is fed into the high precision multimeter (3458A, Agilent) with 10 μΩ sensitivity using four-wire resistance measurement. The measurement process was almost completely automated via a Python program.

In order to determine the performance of our sensor, the digital output code was measured as a function of the temperature from -25 °C to 75 °C with 12.5 °C increment. The measurement rate depends on the selected frequency divisor, see Table 1. Before starting a series of measurements on each test chip, the temperature of the programmable temperature calibrator was set to -25 °C allowing it to stabilize. Every temperature value is set automatically via Python script. After the desired value and stability is reached, the measurement starts. Sensor system and multimeter are read-out almost simultaneously. After 1500 values are collected, temperature increases by 12.5 °C until it reaches 75 °C. A complete evaluation cycle takes about three weeks. It was performed on two arbitrarily selected test chips (chip 3, chip 6).

Table 1: Performance summary of the proposed CMOS digital temperature sensor, taking account of minimum, mean and maximum value of triple standard deviation over the complete test temperature range.

Divisor	fs Hz	$3\sigma_{min}$ °C	$3\sigma_{mean}$ °C	$3\sigma_{max}$ °C
2^{11}	204.8	0.21	1.32	3.75
2^{13}	51.2	0.04	0.61	1.79
2^{15}	12.8	0.03	0.39	0.66
2^{17}	3.2	0.01	0.16	0.54
2^{19}	0.8	0.009	0.10	0.35
2^{21}	0.2	0.008	0.05	0.14
2^{23}	0.05	0.007	0.04	0.06

B. Results

Table 1 shows measured performance data of the proposed sensor system. If the divisor increases, conversion rate (fs) and accuracy decrease accordingly. We use the triple standard deviation of 100 subsequent measurement values for both chips to specify accuracy. Taking the mean value, the temperature accuracy range of our system can be scaled from 1.32 °C to 0.04 °C by changing the divisor.

[1] DAkkS (Deutsche Akkreditierungsstelle) is the national accreditation body of the Federal Republic of Germany. It is legally mandated to carry out accreditations of conformity assessment bodies.

[2] PTB (Physikalisch-Technische Bundesanstalt) is the national metrology institute.

Table 2: Comparison of smart ring-oscillator-based temperature sensors

Resolution	Conversion Rate	Temperature Range	CMOS Technology	Chip Area	Sensor
0.7 °C/LSB	1 Hz	0 °C - 100 °C	80 nm	0.016 mm²	[17]
0.4 °C/LSB	10 Hz	8 °C - 85 °C	0.13 µm	0.04 mm²	[6]
0.18 °C/LSB	100 Hz	5 °C - 100 °C	0.13 µm	0.05 mm²	[7]
0.24 °C/LSB	205 Hz	-25 °C - 75 °C	0.35 µm	0.21 mm²	This
0.0009 °C/LSB	0.8 Hz	-25 °C - 75 °C	0.35 µm	0.21 mm²	Work

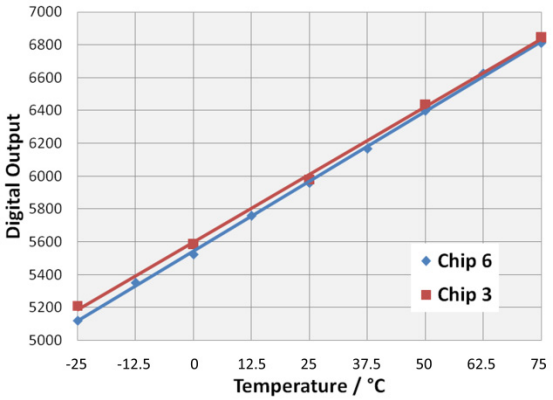


Fig. 6. Digital output code as a function of temperature, measured at 51.2 Hz conversion rate.

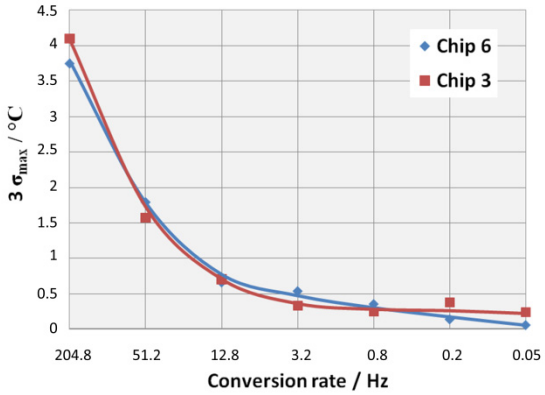


Fig. 7. Accuracy as a function of conversion rate, taking account of the maximum value of the triple standard deviation over the complete temperature range.

With increasing temperature, the digital output code of our system increases nearly linear. Thus, we define resolution by the ratio of maximum temperature difference to the corresponding difference of output codes. Fig. 6 shows the data for both test chips at a conversion rate of 51.2 Hz, reaching a resolution of 0.06 °C. The linearity of the output code as a function of temperature is equal for every suspended divisor while the difference between maximum digital output-codes at 75 °C and minimum digital output code at -25 °C increases. This increase results in a higher resolution of the sensor system.

Fig. 7 demonstrates that the worst case accuracy, hence the maximum value of triple standard deviation, decreases with the conversion rate, reaching 0.06 °C getting a measured value every 20 s. Likewise, the temperature resolution of the proposed sensor system scales with conversion rate from 0.24 °C to 55 µ°C. In Fig. 8, the relationship between the resolution and the conversion rate of the ring-oscillator-based sensor is presented.

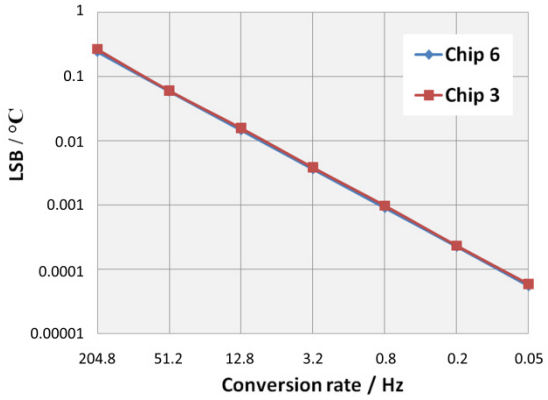


Fig. 8. Resolution of the least significant bit (LSB) as a function of the selected conversion rate.

IV. Conclusion

A CMOS temperature sensor fabricated in 0.35 µm CMOS technology featuring a scalable resolution has been presented.

The sensor can be two-point calibrated at the two ending temperatures with first-order curve fitting because of the linearity of the sensor output code from -25 °C to 75 °C.

A scalable resolution from 0.24 °C to 55 µ°C is achieved through smart evaluation of two different ring oscillator output signals. A comparison of the performance data of the proposed temperature sensor with other smart ring-oscillator-based temperature sensors is presented in Table 2. This table shows that, besides scalable resolution, our sensor concept offer high resolution related to conversion rate. In our future

work we will focus on a fabrication process with smaller feature size in order to minimize chip area.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support of the Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst in the context of the Forschungsschwerpunkt Intelligente Sensorik at University of Applied Sciences, Aschaffenburg, Germany. Technical support of WIKA Alexander Wiegand SE & Co. KG, Klingenberg, Germany, is also greatfully acknowledged.

REFERENCES

[1] G. Roberts and M. Ali Bakhshian, "A Brief Introduction to Time-to-Digital and Digital-to-Time Converters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 3, pp. 153–157, 2010.

[2] J. Zeng, Y. Li, H. Xie, and J. Wen, "CMOS digital integrated temperature sensor," in *6th International Conference on ASIC, 2005. ASICON 2005,* 2005, pp. 248–252.

[3] M. Pertijs, A. Bakker, and J. Huijsing, "A high-accuracy temperature sensor with second-order curvature correction and digital bus interface," in *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001*, 2001, pp. 368–371.

[4] C. C. Chun, L. an Wei, C. C. Yu, and C. Poki, "An accurate CMOS time-to-digital-converter-based smart temperature sensor with negative thermal coefficient," in *Sensors, 2005 IEEE*, 2005, pp. 4.

[5] A. Brönner, D. Fuchs, and U. Brunsmann, "An experimental test chip for TDC-based digital sensors," in *Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg*, Reutlingen, 2010, pp. 67–78.

[6] P. Sunghyun, M. Changwook, and H. C. Seong, "A 95nW ring oscillator-based temperature sensor for RFID tags in 0.13 µm CMOS," in *2009. IEEE International Symposium on Circuits and Systems, 2009. ISCAS*, 2009, pp. 1153–1156.

[7] S. W. Sung, H. L. Jung, and C. SeongHwan, "A ring oscillator-based temperature sensor for U-healthcare in 0.13 µm CMOS," in *2009 International SoC Design Conference* (ISOCC), 2009, pp. 548–551.

[8] K. Kisoo, L. Hokyu, J. Sangdon, and K. Chulwoo, "A 366kS/s 400uW 0.0013mm2 frequency-to-digital converter based CMOS temperature sensor utilizing multiphase clock," in *Custom Integrated Circuits Conference, 2009. CICC '09*. IEEE, 2009, pp. 203–206.

[9] A. Bakker and J. Huijsing, "Micropower CMOS temperature sensor with digital output*," IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 933–937, 1996.

[10] K. Oonchom, B. Arnuttinanon, and O. Wongwirat, "A development of hybrid temperature recorder monitoring system," in *2010 International Conference on Control Automation and Systems (ICCAS)*, 2010, pp. 271–275.

[11] C. Boano, M. Lasagni, K. Römer, and T. Lange, "Accurate Temperature Measurements for Medical Research Using Body Sensor Networks," in *2011 14th IEEE International Symposium on Object/ Component/ Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, 2011, pp. 189–198.

[12] N. H. E. Weste and D. M. Harris, *Integrated circuit design,* 4th ed. Boston, Singapore: Pearson, 2011.

[13] S. M. Sze and K. K. Ng, *Physics of semiconductor devices,* 3rd ed. New York: John Wiley & Sons, 2007.

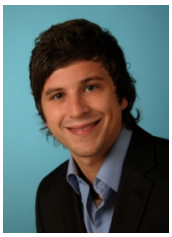
[14] R. J. Baker, *CMOS: Circuit design, layout, and simulation,* 2nd ed. Hoboken, NJ: Wiley-Interscience [u.a.], 2008.

[15] C. C. Chen, P. Chen, C. S. Hwang, and W. Chang, "A Precise Cyclic CMOS Time-to-Digital Converter With Low Thermal Sensitivity," *IEEE Transactions on Nuclear Science*, vol. 52, no. 4, pp. 834–838, 2005.

[16] W. Liu, *Mosfet models for spice simulation, including BSIM3v3 and BSIM4*. New York: Wiley-Interscience Publication, 2001.

[17] K. K. Chan, S. K. Bai, G. L. Chil, and H. J. Young, "CMOS temperature sensor with ring oscillator for mobile DRAM self-refresh control," in *2008. IEEE International Symposium on Circuits and Systems, 2008. ISCAS*, 2008, pp. 3094–3097.

Andreas Brönner received the B. Eng. degree in electrical engineering from university of applied sciences Aschaffenburg, Germany, in 2010 where he has been working towards the M. Eng. degree in the Laboratory of electronic devices. His research interests include high-resolution time-to-digital converters and high-accuracy smart temperature sensors.



Daniel Fuchs received the B. Eng. degree in electrical engineering from university of applied sciences Aschaffenburg, Germany, in 2010 where he has been working towards the M. Eng. degree in the Laboratory of electronic devices and chip design. His research interests include integrated sensor application and high-accuracy smart temperature sensors.

Ulrich Brunsmann is a professor for Electronic Devices and Computational Intelligence at the University of Applied Sciences, Aschaffenburg. He was with Battelle-Institut, Frankfurt, before, where he headed the development of sensors and electronics for space astronomy. He received his PhD (1977) and his diploma degree (1972) in the field of solid state surface physics at the University of Gießen.

Hochschule Ulm

Ein CMOS-Leistungsverstärker für niedrige Versorgungsspannungen

Lukas Schumm, Gerhard Forster

***Zusammenfassung*—In der vorliegenden Arbeit wird die Entwicklung einer linearen Leistungsstufe zum Treiben einer niederohmigen Ausgangslast vorgestellt. Sie wurde in einer 0,35 µm-CMOS-Technologie mit der Versorgungsspannung 3,3 V realisiert und erreicht eine Ausgangsspannung von 3,0 Vpp an 50 Ohm. Die Ruhestromaufnahme von 500 µA ist bezüglich Temperatur und Versorgungsspannung stabilisiert. Der IP-Block beinhaltet eine Rail-to-Rail-Endstufe und eine geeignete Eingangsstufe zur Rückkopplung als Buffer. Die Schleifenverstärkung beträgt 82 dB, die Transitfrequenz liegt über 10 MHz und die Offsetspannung unter 2 mV. Zu erwartende Fertigungstoleranzen wurden im Rahmen einer Entwurfsoptimierung berücksichtigt.**

***Schlüsselwörter*—CMOS-Leistungsverstärker, Arbeitspunktstabilisierung, Entwurfszentrierung, Rail-to-Rail-Buffer.**

I. Einleitung

Der weiterhin ungebrochene Trend zur Verringerung der Strukturgrößen in der Mikroelektronik macht den Entwurf von leistungsfähigen Analogschaltungen zunehmend schwieriger. Während die steigenden Ausgangsleitwerte der Transistoren und die Variabilität der Bauelemente lediglich verhindern, dass die weitere Miniaturisierung effizient genutzt werden kann, stellt die Verringerung der Versorgungsspannung für viele Anwendungen ein ernstes Machbarkeitsproblem dar. Batteriebetriebene Systeme der Kommunikations-, Medizin- oder Automatisierungstechnik profitieren zwar von niedrigen Versorgungsspannungen, dennoch sind auch hier niederohmige Aktoren wie Lautsprecher, Laser oder Ultraschallgeber zu treiben [1]. Hierzu werden lineare Endstufen mit hohem Aussteuerbereich und geringer Offsetspannung benötigt.

Lukas Schumm, lschumm@mail.hs-ulm.de und Gerhard Forster, forster@hs-ulm.de sind Mitglieder der Hochschule Ulm, Eberhard-Finckh-Straße 11, 89075 Ulm

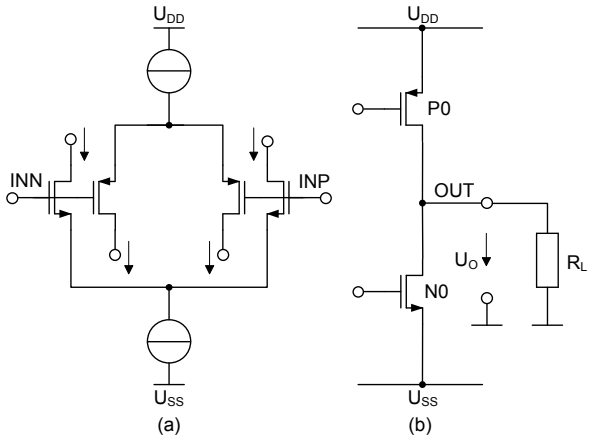


Abbildung 1: Rail-to-Rail-Prinzip am Eingang (a) und Ausgang (b)

Ziel der vorliegenden Arbeit war die Entwicklung eines Leistungsverstärkers als Standardzelle zum Betrieb eines Lastwiderstands R_L = 50 Ω mit einer Wechselsignalleistung P_L = 22,5 mW im Frequenzbereich bis f = 1 MHz und einer Offsetspannung U_{os} < 2 mV, bei einer Ruhe-Leistungsaufnahme P_{Cons} < 2 mW. Die Realisierung sollte in einer CMOS-Standardtechnologie mit der Versorgungsspannung U_B = 3,3 V erfolgen.

Die geforderte Zielspezifikation macht eine Scheitelspannung $\hat{u}_o$ = 1,5 V und damit eine Rail-to-Rail-Endstufe erforderlich. Zur Erfüllung der Linearitäts- und Offsetanforderung wird in diesem Fall zusätzlich eine Vorverstärkerstufe benötigt, damit die Endstufe in eine Rückkopplung eingebettet werden kann [2]. Angesetzt wird eine Open-Loop-Verstärkung V_u = 80 dB bei angeschlossenem Lastwiderstand. Um die Leistungsaufnahme gegenüber Schwankungen der Versorgungsspannung und Temperatur zu stabilisieren, wird zusätzlich eine Versorgungseinheit benötigt. Die hohe, vom Halbleiterhersteller spezifizierte Variabilität der Prozessparameter macht es erforderlich, dass diese bereits bei der Dimensionierung der Transistoren Berücksichtigung findet. Dies soll mit Hilfe eines Optimierungstools unter Beobachtung der zu erwartenden Gesamtausbeute erfolgen.

Im nun folgenden Kapitel 2 wird, ausgehend von den grundlegenden Prinzipien, das Schaltungskonzept erarbeitet. Ebenfalls wird der Weg zu einer ersten Dimensionierung der Bauelemente erläutert. Kapitel 3

zeigt Simulationsergebnisse dieser Vordimensionierung und Kapitel 4 den Weg zur Feindimensionierung unter dem Einsatz des Optimierungstools WiCkeD. Hierbei wird der Einfluss der Variabilität, sowohl hinsichtlich der Technologieparameter als auch der Geometriegrößen, in die Dimensionierung einbezogen [3]. Kapitel 5 zeigt schließlich die Hardware-Realisierung und stellt erste Messergebnisse dar. Kapitel 6 gibt eine kurze Zusammenfassung und endet mit dem Ausblick.

II. Schaltungskonzept

Eine Rail-to-Rail-Endstufe ist nur durch den Betrieb der Endstufentransistoren in Source-Schaltung möglich (Abbildung 1b). Die Ansteuerung dieser Transistoren und die Stabilisierung des Arbeitspunktes im AB-Betrieb sind aufwändiger als mit den klassischen Drain-Schaltungen. Außerdem resultiert daraus Spannungsverstärkung, die eine starke Abhängigkeit von der zu treibenden Last R_L und dem Momentanwert der Signalspannung zeigt und damit zu Verzerrungen führt. Aus diesem Grund ist ein zusätzlicher Differenzverstärker erforderlich, so dass der gesamte Verstärker als rückgekoppelter Buffer betrieben werden kann. Die gesamte Signalspannung wirkt in diesem Fall als Gleichtaktsignal am Eingang des Differenzverstärkers, weshalb auch dieser Rail-to-Rail-Eigenschaft aufweisen muss (Abbildung 1a).

A. Rail-to-Rail-Eingangsstufe

Damit die Eingangs-Gleichtaktspannung den gesamten Bereich zwischen U_{DD} und U_{SS} abdeckt, kommt das bekannte Prinzip der kombinierten P- und NMOS-Differenzstufe zum Einsatz [4]-[7]. Zu beachten ist, dass der Bereich, in dem keiner der beiden Transistoren arbeitet (die sogenannte Dead Zone), vermieden wird. Das gesamte Eingangssignal wirkt als Gleichtaktspannung, was die Forderung nach hoher Gleichtaktunterdrückung zur Folge hat. Für die Stromspiegel sind daher Kaskodestromspiegel vorzusehen, die jeweils zwei im Sättigungsbereich betriebene Transistoren in Serie geschaltet beinhalten. Ihr minimaler Spannungsbedarf beträgt $2U_{DS,sat}$. Der Eingangs-Gleichtaktspannungsbereich wird dann bestimmt durch:

$$U_{igl,\min} = U_{SS} + 2U_{DS,sat} + U_{GS} \tag{1}$$

$$U_{igl,\max} = U_{DD} - 2\left|U_{DS,sat}\right| - \left|U_{GS}\right| \tag{2}$$

Die Schnittstelle der Rail-to-Rail-Differenzstufe zur Endstufe bilden jeweils zwei differentielle Stromausgänge.

B. Rail-to-Rail -Ausgangsstufe

Die Ausgangsspannung der Endstufe nach Abbildung 1b wird begrenzt durch

$$U_{o,\max} = U_{DD} - \left|U_{DS,\min}\right| \tag{3}$$

und

$$U_{o,\min} = U_{SS} + U_{DS,\min} \tag{4}$$

Die Transistoren befinden sich in diesem Fall im Triodenbereich. Die minimale Drain-Source-Spannung $U_{DS,min}$ ist abhängig vom Lastwiderstand R_L. Sie lässt sich bei Annahme maximaler Ansteuerspannung am Gate $U_{GS} \approx U_{DD} - U_{SS}$ wie folgt berechnen:

$$U_{DS,\min} = U_B - U_{TH} + \frac{1}{\beta R_L} - \frac{1}{\beta}\sqrt{\left(\beta\left(U_B - U_{TH}\right) + \frac{1}{R_L}\right)^2 - 2\beta\frac{U_B}{R_L}} \tag{5}$$

wobei

$$\beta = k'\frac{W}{L} \tag{6}$$

und

$$U_B = U_{DD} - U_{SS} \tag{7}$$

Im Gegensatz zum CMOS-Inverter, bei welchem die Gateanschlüsse der Transistoren miteinander verschaltet sind und direkt angesteuert werden können, benötigen die Endstufentransistoren für den AB-Betrieb eine Ansteuerschaltung.

C. Ansteuerung der Endstufe

Die Grundschaltung zur Ruhestromeinstellung wird in Abbildung 2a aufgezeigt. Die Gate-Source-Spannung des Endstufentransistors N0 wird über die Gate-Source-Spannung der Transistoren N13, N11 und N4 festgelegt:

$$U_{GS0} = U_{GS13} + U_{GS11} - U_{GS4} \tag{8}$$

Da N0 ohne Signalstrom I_{Sig} im Sättigungsbereich betrieben wird, bestimmt sich damit der Ruhestrom zu

$$I_{RUHE} = \frac{k_n'}{2}\left(\frac{W}{L}\right)_0 \left(U_{GS0} - U_{THn}\right)^2 \tag{9}$$

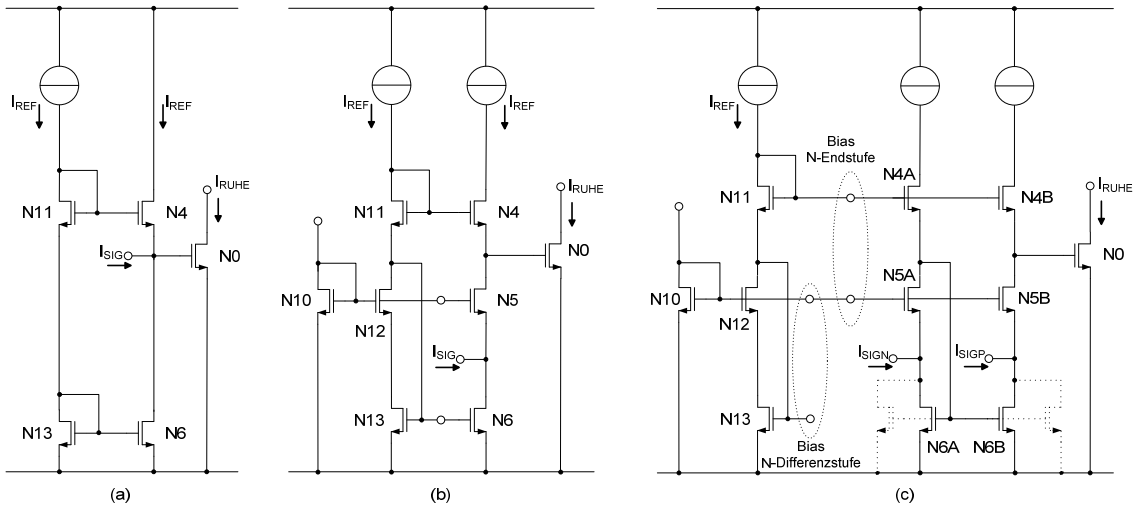


Abbildung 2: Schaltungsevolution zur Endstufe: (a) Grundschaltung, (b) Low Voltage Kaskode, (c) symmetrische Erweiterung

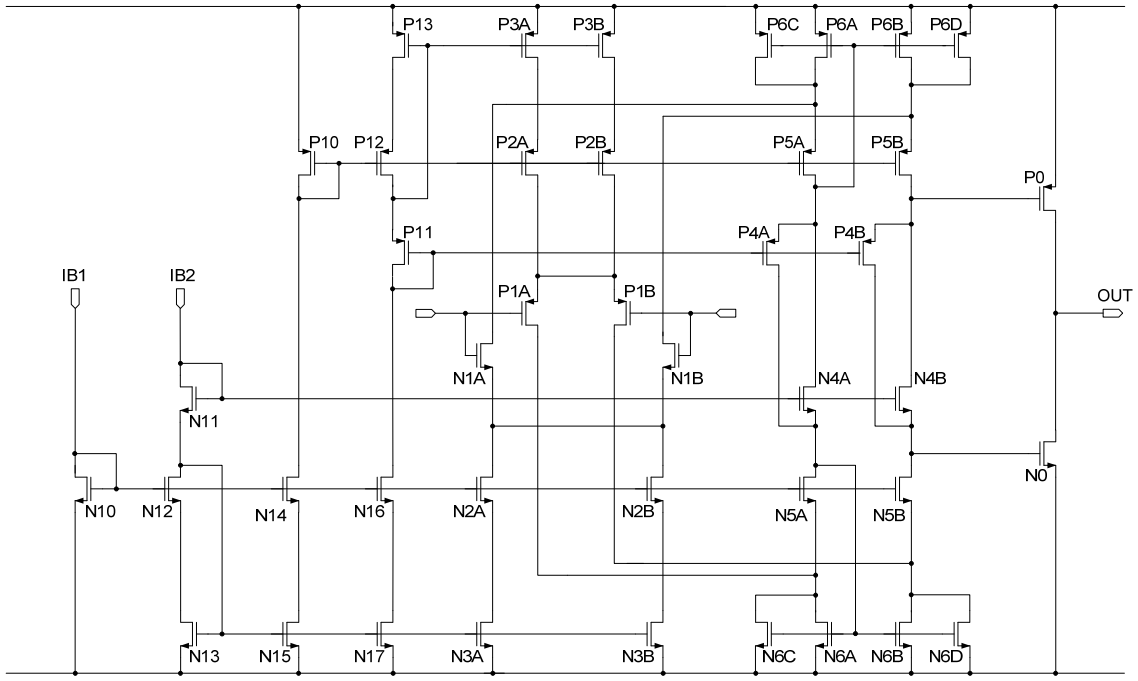


Abbildung 3: Gesamtschaltung des Verstärkerkerns

I_{RUHE} ist also über I_{REF} und die W/L – Verhältnisse der Transistoren eindeutig festgelegt. Um ausreichende Spannungsverstärkung zu ermöglichen, muss der Stromeingang hochohmig sein. Die Kleinsignal-Gate-Source-Spannung von N0 wird aber durch die Source-Impedanz von N4 bestimmt:

$$u_{gs0} \approx \frac{1}{g_{m4}} i_{sig} \qquad (10)$$

Die Transimpedanz u_{gs0}/i_{sig} ist damit zu klein für eine ausreichende Spannungsverstärkung des Gesamtverstärkers. Sie muss deshalb, wie in Abbildung 2b gezeigt, über den Einsatz der Kaskodetransistoren N10, N12 und N5 erhöht werden. Der Knoten zur Einspeisung von i_{sig} bleibt nun weiterhin niederohmig, aber die Kleinsignal-Gate-Source-Spannung des Transistors N0 ergibt sich nun näherungsweise zu:

$$u_{gs0} = r_{05}\left(1 + g_{m5} r_{06}\right) i_{sig} \qquad (11)$$

Um weiterhin den Ruhestrom über I_{REF} und die W/L-Verhältnisse zu fixieren, bedarf es weiterhin des Stromspiegels N11, N4 und einer zusätzlichen Stromquelle I_{REF}, um die Niederohmigkeit am Source-Knoten von N4 aufzuheben. Für den Betrieb bei niedriger Versorgungsspannung ist die Kaskodeschaltung als sogenannte Low-Voltage-Kaskode ausgeführt

(Verbindung des Gate-Anschlusses von N13 mit dem Drain-Anschluss von N12). Für eine symmetrische Signaleinkopplung, wie sie von der Differenzstufe geliefert wird, ist diese Version allerdings noch nicht nutzbar. Daher wird die Schaltung um eine symmetrische Kaskode erweitert (Abbildung 2c). Die Signaleinkopplung ist jetzt für beide Eingangsströme i_{sign} und i_{sigp} möglich. Die zusätzlichen Bias-Spannungen in der Schaltungsanordnung können unter anderem für die Differenzstufe genutzt werden. Infolge des Gleichstromanteils im Signalstrom muss der Transistor N6 vergrößert, z.B. verdoppelt werden. Zum Betrieb der positiven Halbwelle muss dieselbe Schaltung noch um ihr Spiegelbild, ausgeführt mit PMOS-Transistoren, ergänzt werden (Abbildung 3). Die Stromquellen sind dabei durch floatende Stromquellen (N4, P4) zu ersetzen. Nach Einfügen der Differenzstufe und weiterer Transistoren zur Festlegung der Bias-Spannungen ist die Konzeptphase des Verstärkerkerns abgeschlossen [8]. Der Verstärkerkern wird mit zwei Bias-Strömen I_{B1} und I_{B2} betrieben.

D. Vordimensionierung des Verstärkerkerns

1) Endstufentransistoren

Um eine hohe Spannungsverstärkung zu erreichen, wird eine große Transkonduktanz angestrebt. Da die Exemplarstreuung der Endstufentransistoren für die Offsetspannung unerheblich ist, können diese mit der minimalen Kanallänge L = 0,35 µm ausgeführt werden. Die gewünschte Scheitelspannung $\hat{u}_o$ = 1,50 V macht bei U_{SS} = -1,65 V nach Gl. 4 eine minimale Drain-Source-Spannung $U_{DS,min}$= 150 mV am Transistor N0 erforderlich. Mit dem Lastwiderstand R_L = 50 Ω resultiert dann aus Gl. 5 und 6 eine Mindestkanalweite W = 1100 µm.

2) Stromspiegel

Um das Verhältnis von Ruhestrom in der Endstufe zum Ruhestrom in der restlichen Schaltung in einem vernünftigen Maß zu halten, sollte die Summe der einzelnen Drainströme nicht größer sein als der Drainstrom von N0. Der Strom durch die Differenzverstärker muss aber für eine hohe Bandbreite ausreichend groß sein. Um Abweichungen in den physikalischen Eigenschaften von identisch ausgelegten Transistoren zu vermeiden, sollten diese nach Möglichkeit gleich groß gewählt werden. Im einfachsten Fall ist

$$U_{GS13} = U_{GS0} \ , \qquad (12)$$

was zu einer einheitlichen Gate-Source Spannung der Transistoren N13, N15, N17, N3, N6 bzw. N12, N14, N16, N2, N5 und Drain-Source Spannung der Transistoren N13, N15, N17, N3, N6 bzw. N12, N5 führt. Unter diesen Voraussetzungen und hinsichtlich des Ausgangswiderstandes, dem Ziel des Betriebs im Sättigungsbereich und einem gewählten Drainstrom für alle Transistoren in Höhe von 15 µA wurden die Werte für Kanalweite und -länge festgelegt (W = 20 µm, L = 2 µm).

3) Kaskodetransistoren

Bei den Kaskodetransistoren N12, N14, N16, N2 und N5 ist nach Gl. 11 neben dem Ausgangswiderstand r_o auch die Transkonduktanz g_m von Bedeutung, während die Offsetspannung eine untergeordnete Rolle spielt. Ihre Dimensionierung wird daher auf W/L = 20 µm/1 µm angesetzt. Mit der Diodenschaltung N10 wird die Bias-Spannung $U_{GS10} = U_{DS13,sat} + U_{GS12} \approx$ 900 mV erzeugt. Mit der Wahl $I_{B1} = I_{B2}$ = 15 µA und L_{10} = 1 µm ergibt sich daraus die Kanalweite W_{10} = 1,2 µm.

4) Floatende Stromquellen

Die Transistoren P4A, N4A und P4B, N4B stellen jeweils eine floatende Stromquelle dar. Wenn sich unterhalb der Stromquelle die Spannung ändert, bleibt der Strom dennoch gleich. Das gleiche gilt für den Knoten oberhalb der Stromquellen. Da jeder einzelne Transistor P4A bzw. N4A nur die Hälfte des Stroms von N5A führt, wurde die halbe Kanalweite W_4 = 10 µm festgelegt.

5) Differenzstufentransistoren

Als Kompromiss zwischen den Zielgrößen: hohe Bandbreite, hohe Transkonduktanz, geringe Offsetspannung und geringe Fläche wurde die Kanallänge auf L = 1,5 µm angesetzt. Mit einem geringen Overdrive von ΔU_{GS} = 50 mV für eine hohe Verstärkung resultiert die Kanalweite W = 100 µm. Die Verstärkung setzt sich aus zwei Teilen zusammen. Die Steilheit der Differenzstufentransistoren multipliziert mit dem Widerstand am Gate-Knoten von N0 bildet den ersten Term und die Steilheit des Endstufentransistors N0 multipliziert mit dem Lastwiderstand R_L den zweiten:

$$V_u = (-g_{m1}(r_{05}(1 + g_{m5}r_{06})))(-g_{m0}R_L) \qquad (13)$$

Die gleiche Verstärkung kommt noch einmal hinzu, wenn die PMOS-Transistoren symmetrisch zu den NMOS-Transistoren dimensioniert werden. Den PMOS-Transistoren wurde bei gleicher Kanallänge die jeweils doppelte Kanalweite gegeben. Die Biasströme I_{B1} und I_{B2} werden von einer Versorgungseinheit geliefert, welche im Folgenden beschrieben wird.

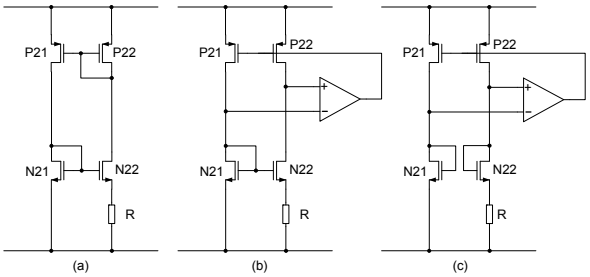


Abbildung 4: Schaltungsevolution der Versorgungseinheit: Grundschaltung (a), Erweiterung um Regelkreis (b), Eliminierung des dominanten Pols (c)

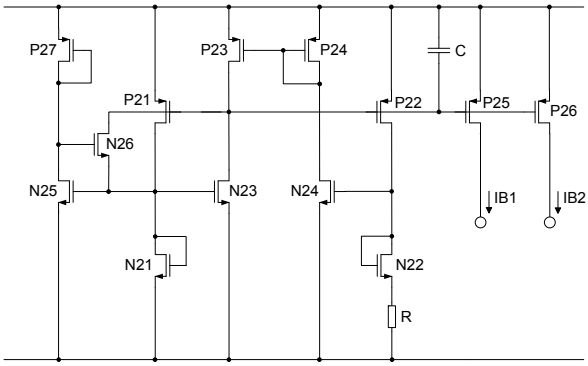


Abbildung 5: Gesamtschaltung Versorgungseinheit

E. Entwicklung der Versorgungseinheit

Den Ansatz zur Versorgungseinheit bildet der Beta-Multiplizierer (Abbildung 4a). Er liefert einen konstanten Strom abhängig vom Absolutwert *R*, jedoch unabhängig von Temperatur und Versorgungsspannung, sofern R aus Teilen mit positiven und negativen Temperaturkoeffizienten zusammengesetzt ist [8]. Der Strom durch N22 wird dabei bestimmt durch die W/L-Verhältnisse von N22 und N21 zueinander:

$$I_{D22} = \frac{2}{R^2 k_n' \left(\frac{W}{L}\right)_{21}} \left(1 - \frac{1}{\left(\frac{W}{L}\right)_{22} \Big/ \left(\frac{W}{L}\right)_{21}}\right)^2 \quad (14)$$

Allerdings besitzt die Schaltung in dieser Form eine Nichtidealität. Unterschiedliche Drain-Source-Spannungen an den PMOS-Transistoren führen zu einem Fehler im Stromspiegel. Durch einen Regelkreis mit Hilfe eines OTAs[1], dessen Ausgang an die Gateanschlüsse der PMOS-Transistoren geführt wird, kann dem entgegen gewirkt werden (Abbildung 4b).

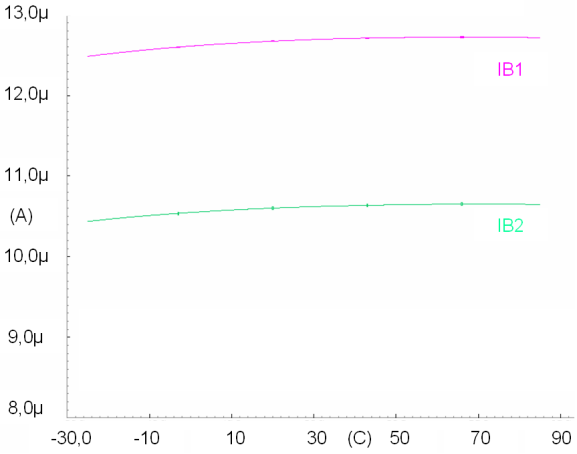


Abbildung 6: Biasströme in Abhängigkeit von der Temperatur

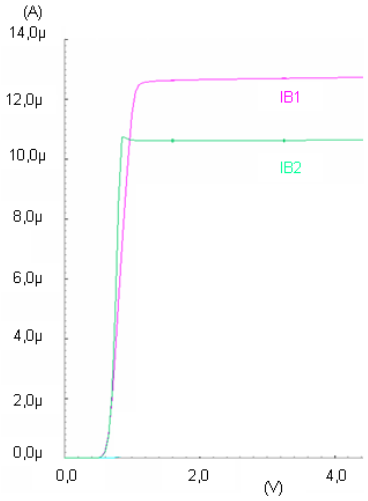


Abbildung 7: Biasströme in Abhängigkeit von der Versorgungsspannung

Dabei entstehen jedoch zwei dominante Pole, einer am Ausgang und einer am nichtinvertierenden Eingang des OTAs. Die führt zu Schwierigkeiten in der Frequenzgangkompensation. Eliminierung eines dominanten Pols wird erreicht, indem der Transistor N22 als Diode beschaltet wird (Abbildung 4c). Mit dieser Maßnahme ändert sich der DC-Wert nicht. Eine Startschaltung, eine Auskopplung und ein Kompensationskondensator werden noch benötigt, um die Versorgungseinheit zu komplettieren (Abbildung 5).

[1] Operational Transconductance Amplifier

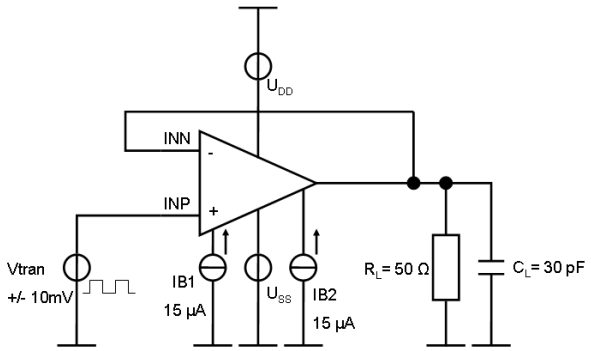


Abbildung 8: Testbench für die Transient- und DC-Analyse

Tabelle 1: Simulationsergebnisse nach der Vordimensionierung

Performance	Spezifikation	Wert
Spannungsverstärkung V_u	> 80 dB	104,8 dB
Ruhestrom Endstufe I_{Ruhe}	< 500 µA	4,5 mA
Max. Ausg.-Spg. $U_{OUT,max}$	> 1,5 V	1,56 V
Min. Ausg.-Spg. $U_{OUT,min}$	< -1,5 V	-1,6 V
Phasenreserve φ_R	> 60°	29°
Leistungsaufnahme P_{cons}	< 2 mW	15,4 mW
Transitfrequenz f_T	> 5 MHz	51 MHz

III. Schaltungssimulation

A. Simulation der Versorgungseinheit

Die Dimensionierung der W/L-Verhältnisse für die Versorgungseinheit soll hier nicht näher beschrieben werden. Die Simulation der beiden Bias-Ströme zeigt die Stabilität gegenüber der Temperatur- und Versorgungsspannungsänderung (Abbildungen 6 und 7).

B. Testbench für die Verstärkersimulation

Für die Transient- und die DC-Analyse muss der Ausgang des Verstärkers auf den invertierenden Eingang zurückgekoppelt werden (Abbildung 8). Der nichtinvertierende Eingang wird mit einer Spannungsquelle angesteuert. Zusätzlich sind noch die Versorgungsspannungen und die von der Versorgungseinheit gelieferten Biasströme anzulegen. Für die AC-Analyse bedarf es einer anderen Beschaltung (Abbildung 9). Der invertierende Eingang wird nicht mit dem Ausgang verbunden, sondern auf Masse gelegt. Der Gleichstromregelkreis sorgt dafür, dass die AC-Analyse im aktiven Bereich, unabhängig von der Offsetspannung, durchgeführt wird. Das Tiefpassfilter ist so ausgelegt, dass es im interessierenden Frequenzbereich nicht in die Verstärkung eingeht.

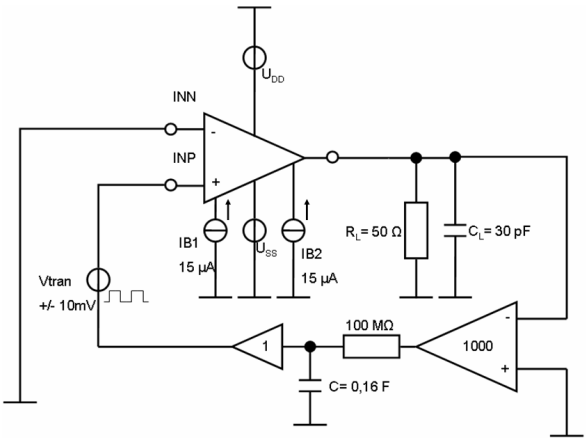


Abbildung 9: Testbench für die AC-Analyse

C. Simulation des Verstärkers

Tabelle 1 zeigt die Simulationsergebnisse des konzeptionierten Verstärkers. Die Verstärkung, die Transitfrequenz und die Aussteuergrenzen sind bereits innerhalb der Spezifikationsgrenzen. Allerdings ist der Ruhestrom viel zu hoch, die Phasenreserve zu niedrig und die Aussteuergrenzen sind unsymmetrisch. Um dem entgegen zu wirken, sind Optimierungsmaßnahmen per Hand denkbar. Das Absenken des Ruhestroms wird durch starke Reduzierung des W/L-Verhältnisses von P4B und N4B gegenüber P4A und N4A erreicht. Durch die Low-Voltage-Kaskode wird allerdings die Drain-Spannung von N5A festgehalten, weshalb das symmetrische Verhältnis von N4A zu N4B aufgegeben werden muss. Symmetrische Aussteuergrenzen werden erreicht, indem die W/L-Verhältnisse der Endstufentransistoren N0 und P0 angepasst werden. Anstelle dieser manuellen Vorgehensweise sollte die weitere Dimensionierung mit Hilfe eines Optimierungstools erfolgen.

IV. Entwurfsoptimierung

A. Optimierungstool WiCkeD

Dieses Optimierungstool der Fa. MunEDA wurde bereits in [9] vorgestellt. Es verfügt über drei Optimierungsarten. Die Feasibitity Optimization konzentriert sich auf die Optimierung hinsichtlich Nebenbedingungen. Am Beispiel einer N-Differenzstufe sind das die Bedingungen des Betriebs in Inversion, einer bestimmten Mindestfläche, welche den Mismatch durch die Transistoreinsatzspannung begrenzt und die Forderung nach gleicher Kanallänge und –weite. Die Optimierung in Bezug auf die Performance (sog. Nominal Optimization) führt die Kenndaten über mehrere Iterationsschritte in die vorgegebenen Spezifikationen hinein. Die rechenintensivste Optimierung richtet ihr Augenmerk auf die Verbesserung der Ausbeute (Yield Optimization).

Tabelle 2: Simulationsergebnisse nach der Nominal Optimization

Performance	Spezifikation	Wert
Spannungsverstärkung V_u	> 80 dB	82,4 dB
Ruhestrom Endstufe I_{Ruhe}	< 500 µA	337,6 µA
Max. Ausg.-Spg. $U_{OUT,max}$	> 1,5 V	1,58 V
Min. Ausg.-Spg. $U_{OUT,min}$	< -1,5 V	-1,59 V
Phasenreserve φ_R	> 60°	83,8°
Leistungsaufnahme P_{cons}	< 2 mW	1,6 mW
Transitfrequenz f_T	> 5 MHz	13,05 MHz

Tabelle 3: Transitorgrößen nach der Nominal Optimization

Transistor	W (µm)	L (µm)
P1A,B	209	1,6
N1A,B	106	4,8
P4A	29	5
P4B	10	5
N4A	33,4	1,5
N4B	6	1,5
P0	2990	0,36
N0	870	0,35

Durch die Designzentrierung, eine Verschiebung des Designs innerhalb des mehrdimensionalen Spezifikationsraums, wird die Schaltung unempfindlicher gegenüber Parameterstreuungen und Fertigungstoleranzen. Da zur Gesamtoptimierung unterschiedliche Testbenches in einem Lauf nötig sind, wurden diese mit Hilfe von Umschaltern festgelegt.

B. Entwurfszentrierung mit WiCkeD

1) Feasibility Optimization

Um die Nebenbedingungen zu erfüllen, kann zwischen zwei Algorithmen gewählt werden. „Find closest point“ sucht den kürzesten Abstand, um das Design in der Feasibility Region zu platzieren. Der von uns verwendete Algorithmus „find central point“ bringt das Design zusätzlich noch in die Mitte der Feasibility Region. Die Nebenbedingungen konnten durch kleine Änderungen in den W/L-Verhältnissen erfüllt werden. Die Simulationszeit für zwei Iterationsschritte beträgt zwölf Minuten. Alle Transistoren befinden sich im Bereich der Feasibility Region. Damit wurde der bestmögliche Startpunkt für die nachfolgende Nominal Optimization gefunden.

2) Nominal Optimization

Die Optimierung hinsichtlich der Zielspezifikationen konnte alle Werte in die vorgegebenen Grenzen führen (Tabelle 2). Die Simulationszeit für drei Iterationsschritte beträgt eine Stunde. Ein Vergleich der W/L-Verhältnisse mit den Werten aus der Vordimensionierung zeigt, wie die Spezifikationen erreicht wurden. Der Optimierer hat alle W/L-Verhältnisse verändert, aber die maßgeblichen Änderungen sind in der floatenden Stromquelle und den Endstufentransistoren zu finden (Tabelle 3). Die Symmetrie in der floatenden Stromquelle wurde aufgegeben. Dadurch konnte der Ruhestrom auf 337,6 µA reduziert werden. Gleichzeitig verringerte sich die Verstärkung auf 82,4 dB, was aber immer noch innerhalb der Spezifikationsgrenzen liegt.

Mit dem Herabsetzen des Ruhestroms konnte auch die gesamte Leistungsaufnahme auf 1,6 mW gesenkt werden. Die Reserven in der Transitfrequenz konnten gegen eine Erhöhung der Phasenreserve auf 83,3° eingetauscht werden. Die Transitfrequenz ist mit 13,05 MHz immer noch ausreichend weit entfernt von der festgelegten Grenze (> 5 MHz). Bei der Betrachtung der optimierten W/L-Verhältnisse ist allerdings das unübliche Aspect Ratio der Differenzstufen $(W/L)_{P1}/(W/L)_{N1}$ auffällig (~ 6). Nach erfolgreicher Nominal-Optimierung konnte die Ausbeute mit der Monte Carlo-Analyse abgeschätzt und beurteilt werden.

3) Monte Carlo-Analyse

Neben den drei Optimierungsarten besitzt WiCkeD eine Reihe von Analysemöglichkeiten. Mit der Monte Carlo-Analyse kann die Gesamt- und Einzelausbeute simuliert werden. Die Gesamtausbeute in Höhe von 70 % wird durch die Einzelausbeute der Verstärkung (93 %), des Ruhestroms (82 %) und der Offsetspannung (94 %) bestimmt. Die Simulationszeit für 100 Samples beträgt eine Stunde.

4) Yield Optimization

Im Anschluss an die Monte Carlo-Analyse wird durch die Yield Optimization die Ausbeute erhöht. Die Simulationszeit für zwei Iterationsschritte beträgt zwei Tage. Die Ausbeuteoptimierung konzentriert sich auf die Verstärkung und den Ruhestrom. Sie führt das Aspect Ratio der Differenzstufe zurück auf einen vernünftigen Wert von ca. 3 (W = 209 µm, L = 2,8 µm für P1A,B und W = 125 µm, L = 4,7 µm für N1A,B). Die Ausbeute konnte damit um 8 % auf 78 % erhöht werden. Für eine Serienproduktion wäre dieser Wert noch nicht zufriedenstellend. Aus Zeitgründen wurde jedoch sogleich mit der Hardwarerealisierung begonnen.

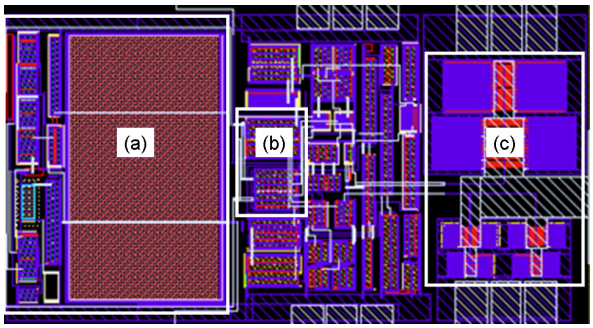


Abbildung 10: Layout (Corebereich): (a) Versorgungseinheit, (b) Differenzstufentransistoren und (c) Endstufe.

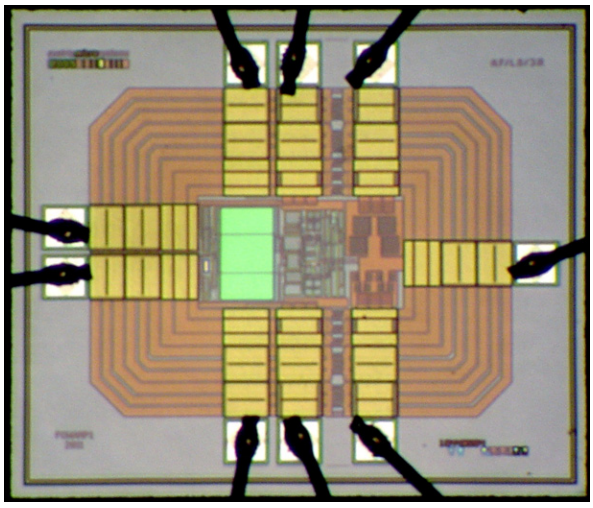


Abbildung 11: Chipfoto

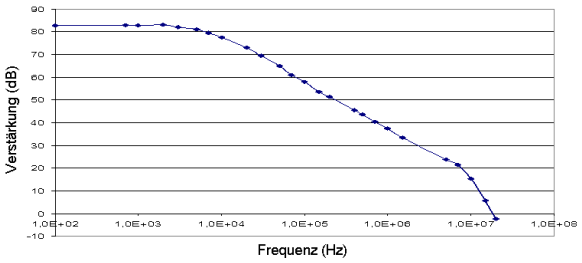


Abbildung 12: Frequenzgang der Verstärkung mit Last, jedoch ohne Rückkopplung

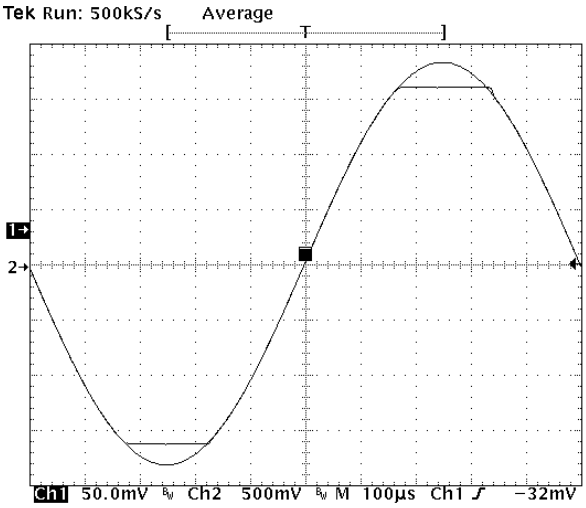


Abbildung 13: Spannungsbegrenzung am Ausgang

V. HARDWAREREALISIERUNG

A. Layoutentwurf

Abbildung 10 zeigt die Aufteilung in die Versorgungseinheit, die Differenzstufe als Quadlayout und die Endstufe. Aufgrund der Wärmeentwicklung der Endstufe wurde diese so weit wie möglich von der Differenzstufe entfernt platziert. Die Stromdichte im Metall zwingt zu einer Aufteilung der Endstufentransistoren. Der PMOS-Transistor wird vierfach, der NMOS-Transistor achtfach angeordnet.

B. Gefertigter Chip

In Abbildung 11 ist der Verstärkerkern mit den Anschlusspins zu sehen mit den beiden Eingängen auf der linken und dem Ausgang auf der rechten Seite. Oben und unten befinden sich die Pins für die Versorgungsspannung der Versorgungseinheit und der Verstärkerstufe und die Leistungspins für die Versorgungsspannung der Endstufe. Zusätzlich sind Pins für die Biasströme I_{B1} und I_{B2} zu Testzwecken herausgeführt. Die Chipfläche beträgt 1190 µm x 1000 µm, die des Core-Bereichs 450 µm x 230 µm.

C. Messungen am Chip

Im Bodediagramm, welches bei einer speziellen Ausgangsgleichspannung U_o = 0,5 V aufgenommen wurde (Abbildung 12), ist zu sehen, dass vor Erreichen der Transitfrequenz ein Pol auftaucht. Dieser führt zur Instabilität des Verstärkers bei Rückkopplung als Unity Gain Buffer. Die Abhängigkeit der Phasenreserve von der Ausgangsspannung, welche in der Simulation nicht ausreichend berücksichtigt wurde, ist der Grund für diesen unerwünschten Effekt. Für die Messung des Aussteuerbereichs musste daher auf den Verstärkungsfaktor 10 zurückgekoppelt werden. Abbildung 13 zeigt die Begrenzung des Aussteuerbereichs bei Ansteuerung mit einem Sinussignal. Abbildung 14 zeigt den Aussteuerbereich als Funktion des Lastwiderstands R_L. Mit derselben Messeinrichtung wurde die Impulsantwort an R_L = 50 Ω aufgenommen (Abbildung 15). Der Übergang im Nulldurchgang ist typisch für diese Art Verstärker, da die Verstärkung der Endstufe vom momentanen Ausgangsstrom abhängt.

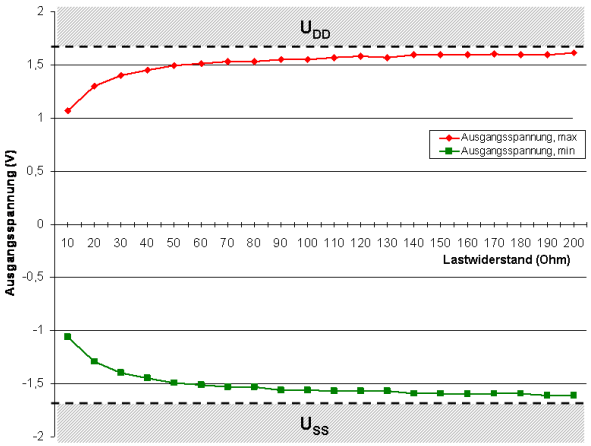


Abbildung 14: Spannungsbegrenzung in Abhängigkeit vom Lastwiderstand R_L

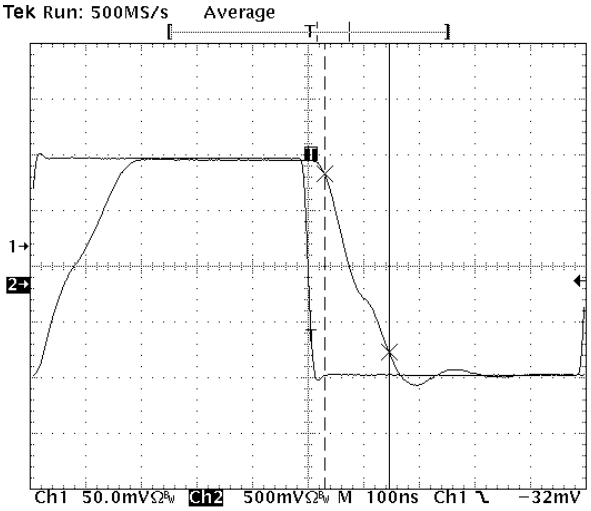


Abbildung 15: Impulsantwort

VI. Zusammenfassung und Ausblick

Für batteriebetriebene Systeme in der Kommunikations-, Medizin- und Sensortechnik wurde eine lineare Leistungsstufe entwickelt und um eine Versorgungseinheit erweitert. Durch den Entwurf einer Testbench konnte die Schaltung mittels WiCkeD optimiert werden. Nach dem Layoutentwurf für den IP-Core wurde der Chip gefertigt, in die aufgebaute Messumgebung eingebettet und verifiziert.

Dabei konnten die wichtigsten Ziele wie Aussteuerbereich, Verstärkung, Offsetspannung, Ruhestrom und Bandbreite erreicht werden. Das Optimierungstool wurde bei der Dimensionierung effizient eingesetzt. Neben der verbesserten Kompensation des Verstärkers zur Frequenzstabilisierung besteht Verbesserungspotential in der weiteren Optimierung mit dem Ziel höherer Ausbeute. Dazu sollen neue WiCkeD Features genutzt werden.

Danksagung

Die Autoren bedanken sich bei Eugen Ringwald und Josef Schäfer für die technische Unterstützung während des gesamten Projekts, bei Alice Frapsauce, Oliver Graf und Franz Gritschneder für ihre Arbeit am Simulator, bei Jens Rechtsteiner für die Erstellung des Layouts und bei Patricia Kruger und Marisa Mitchell für Aufbau und Test.

Literaturverzeichnis

[1] G. Ferri, W. Sansen, „A Rail-to-Rail Constant-gm Low-Voltage CMOS Operational Transconductance Amplifier“ *IEEE Journal of solid-state Circuits*, Vol. 32, No. 10, Oktober 1997.

[2] S. Gierkink et al., “Some Design Aspects of a Two-Stage Rail-to-Rail CMOS Op Amp” *Analog Integrated Circuits and Signal Processing*, Kluwer Academic Publisher, Juli 1998.

[3] H. E. Graeb, “Analog Design Centering and Sizing”, *Springer*, Dordrecht 2007.

[4] T. Stockstad, “A 0.9-V 0.5-µA Rail-to-Rail CMOS Operational Amplifier” *IEEE Journal of solid-state Circuits,* Vol. 37, No. 3, März 2002.

[5] J. H. Huijsing, D. Linebarger, “Low-Voltage Operational Amplifier with Rail-to-Rail Input and Output Ranges” *IEEE Journal of solid-state Circuits,* Vol. Sc-20, No. 6, Dezember 1985.

[6] J. A. Fisher, R. Koch, “A Highly Linear CMOS Buffer Amplifier“ *IEEE Journal of solid-state Circuits,* Vol. Sc-22, No. 3, Juni 1987.

[7] M. D. Pardoen, M. G. Degrauwe, „A Rail-to-Rail Input/ Output CMOS Power Amplifier” *IEEE Journal of solid-state Circuits,* Vol. 25, No. 2, April 1990.

[8] R. J. Baker, „CMOS Circuit Design, Layout, and Simulation“ *IEEE Press*, Piscataway 2005.

[9] F. Mrugalla, G. Vallant, G. Forster, „Dimensionierung und Entwurfszentrierung analoger Schaltungen mit WiCkeD“ *Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg*, Karlsruhe, Juli 2009.

Lukas Schumm ist seit dem Abschluss seines Bachelorstudiums 2010 Wissenschaftlicher Mitarbeiter an der Fakultät Elektrotechnik der Hochschule Ulm. Er befasst sich mit der Schaltungsentwicklung und –optimierung. Er absolviert ein Masterstudium in Systems Engineering and Management, welches er voraussichtlich im Sommersemester 2012 abschließen wird.

Gerhard Forster ist seit 1992 Professor für Elektronik und Mikroelektronische Schaltungen an der Hochschule Ulm. Sein Schwerpunkt liegt auf dem Gebiet des Entwurfs von Mixed-Signal-ASICs. Nach seinem Studium der Physik an der Universität Heidelberg befasste er sich zunächst in einer Forschungsabteilung (heute Daimler Forschungszentrum) mit der Entwicklung und Anwendung neuer Halbleiterprozesse. Er ist Herausgeber des vorliegenden Tagungsbandes.

hochschule aschaffenburg
university of applied sciences

Universeller Sensorsignalkonverter für TDC-basierte digitale Sensorik

Daniel Fuchs, Andreas Brönner, Ulrich Brunsmann

***Zusammenfassung*—Das Konzept eines universellen Sensorsignalkonverters (US^{2}C) auf Basis der Zeit-zu-digital Wandlung ist Thema dieser Veröffentlichung. Die Hauptkomponenten des Systems, ein Signal-zu-Puls Generator (SPC), ein Zeit-zu-Digital Wandler (TDC) sowie ein OnChip Temperatursensor werden auf dem konfigurierbaren CMOS-Chip HAB10 integriert, der in 150nm Technologie gefertigt werden wird. Eine bereits realisierte diskrete analoge Sensoranpassschaltung ermöglicht die Auswahl des Sensoreingangs. Zu den sieben vorkonfigurierten Sensoren gehören Druck- sowie Temperatursensoren. Durch präzise Pulse sensorgrößenabhängiger Dauer, generiert von einem Signal-zu-Puls Wandler und gemessen von einem hochauflösenden TDC, wird in der Simulation bei einer Auflösung von 12-Bit eine Sensormessrate von 2 kHz erreicht. Der OnChip Temperatursensor ist für einen Temperaturbereich von -10 bis 80 °C bei einer Messrate von 500 Hz zur Kompensation temperaturabhängiger Größen ausgelegt.**

***Schlüsselwörter*—CMOS, Verzögerungsstrecke, Signalverarbeitung, Sensorik, konfigurierbare Verzögerung, OnChip Temperatursensor.**

I. Einleitung

Die Wandlung von Sensorsignalen ist ein unverzichtbarer Bestandteil der Prozessindustrie. Sensor-Messumformer, basierend auf der herkömmlichen Analog-zu-Digital Wandlung (ADC) überfluten den Markt. Mit Beginn der 80er Jahre wurden neben mechanischen Messumformern, die ersten elektronischen Messumformer entwickelt. Hierbei stellten elektronische Druckmessumformer [1] den Anfang einer rasanten Entwicklung in der elektronischen Sensorsignalverarbeitung dar. Heute verfügen elektronische Transmitter über eine kundenspezifische Online-Konfiguration via Software, erzeugen als Ausgangssignal meist ein normiertes analoges elektrisches Einheitssignal von 4 mA bis 20 mA und kommunizieren mit der Außenwelt per Bluetooth oder HART-Protokoll. Elektronische Messumformer finden Anwendung in zahlreichen Industriesektoren, dazu gehören:

- Energietechnik
- Erneuerbare Energien
- Automotive
- Lebensmittelbranche
- Pharmaindustrie
- Maschinenbau.

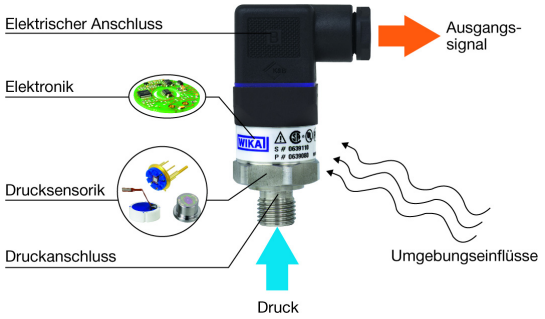


Abbildung 1: Beispielhafter Aufbau eines Drucktransmitters. (Quelle: WIKA, Alexander Wiegand SE & Co. KG)

Mehr als 60% aller industriell erfassten Sensordaten stammen von Druck- bzw. Temperatursensoren. Die zur Konvertierung verwendeten Messumformer verfügen in der Regel über eine herkömmliche A/D Wandlung des Sensorsignals sowie über eine digitale Signalverarbeitung mit Hilfe von Mikrocontrollern (µC). In Abbildung 1 ist beispielhaft der Aufbau eines aktuellen Drucktransmitters dargestellt. Die technische Herausforderung liegt bei heutigen Entwicklungen vor allem im knappen Energiehaushalt dieser Produkte, der Stromverbrauch der gesamten Elektronik darf maximal bei ca. 3 mA liegen.

Der hier vorgestellte universelle Sensorsignalkonverter auf Basis der Zeit-zu-Digital Wandlung [2] stellt ein Konzept dar, welches neue Wege in der Digitalisierung von Sensorsignalen sowie der Signalverarbeitung aufweist. Mit dem Ziel, zukünftig einen universellen Sensorsignal-ASIC einzusetzen, ist diese Arbeit ein erster Schritt hinsichtlich einer solchen Entwicklung. Die Faktoren Stromverbrauch, Messrate, Kosten, Platzbedarf und Wiederverwendbarkeit spiel-

Daniel Fuchs, daniel.fuchs@h-ab.de / d.fuchs@wika.de, Andreas Brönner, andreas.broenner@h-ab.de und Ulrich Brunsmann, ulrich.brunsmann@h-ab.de, sind Mitarbeiter der Hochschule Aschaffenburg, Fakultät Ingenieurwissenschaften, Labor für elektronische Bauelemente und Chipdesign, Würzburger Straße 45, 63743 Aschaffenburg.

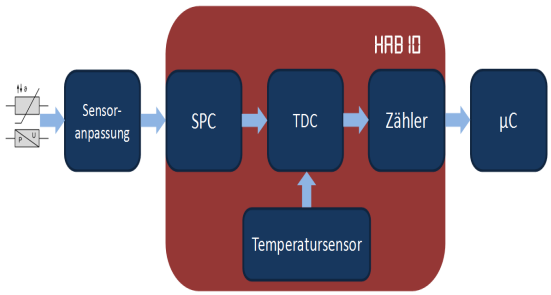


Abbildung 2: Blockschaltbild des universellen Sensorsignalkonverters.

ten bei der Konzeptionierung dieses Systems eine ausschlaggebende Rolle. Ausgelegt ist der US^{2}C in erster Linie für Druck- und Temperatursensoren [3], weitere Sensorarten können angepasst werden.

Im Folgenden wird zunächst in Abschnitt II die Funktionsweise des Gesamtsystems beschrieben, welches in Abbildung 2 schematisch dargestellt ist. Zudem wird in diesem Abschnitt die Entwicklungsumgebung vorgestellt. Daraufhin gehen wir in Abschnitt III auf Aufbau und Funktion der Einzelkomponenten ein. In Abschnitt IV wird der Messablauf des Systems im Detail behandelt. Die Simulationsergebnisse werden in Abschnitt V präsentiert. Es folgt ein Fazit sowie ein Ausblick auf zukünftige Arbeiten.

II. Aufbau und Funktionsweise des universellen Sensorsignalkonverters

Der US^{2}C stellt zum Einen im Sensorsignalpfad das Ergebnis einer Sensorsignalwandlung digital am Ausgang zur Verfügung. Zum Anderen ist das System dazu ausgelegt, die OnChip Temperatur [4] digital auszugeben. Abbildung 2 zeigt das Blockschaltbild der einzelnen Systemkomponenten. Sensorsignale am Eingang des US^{2}C werden von der diskret realisierten Sensoranpassschaltung in einen definierten Steuerspannungsbereich von 0,6 bis 1,5 V angehoben. Dies geschieht durch externe Beschaltung eines Instrumentenverstärkers. Die Sensoranpassung dient somit als Schnittstelle zwischen den Sensoren am Eingang des US^{2}C und der Eingangsstufe des SPC auf dem HAB10. Steuerspannungsabhängige Ausgangspulse, deren Dauer von dem zu messenden Sensorsignal abhängt, werden vom SPC generiert. Der sensorgrößenabhängige Anteil an der Dauer des Ausgangspulses beträgt maximal 30 ns, dessen Zeitoffset ist durch Konfiguration flexibel einstellbar. Ebenfalls durch Konfiguration kann der SPC in einen linearen OnChip Temperatursensor für einen Temperaturbereich von -10 bis 80 °C umgewandelt werden. Anders als bei der Auswertung des Sensorsignals sind die Ausgangspulse hierbei temperaturabhängig. Der zyklische Pulsschwund TDC [5] verkürzt den sensorgrößenabhängigen Eingangspuls zyklisch um einen als Pulsschwund bezeichneten Betrag. Die Gesamtzahl der Durchläufe, welche benötigt werden, um den Eingangspuls zu eliminieren, wird von einem nachfolgenden Zähler erfasst. Der Zählerstand enthält somit die Information über die Messgröße bzw. die OnChip Temperatur. Die Auswertung des Zählerstandes und die anwendungsspezifische Konfiguration des Systems gehören zu den Aufgaben des nachgeschalteten µC.

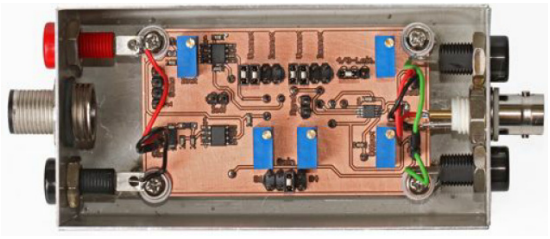


Abbildung 3: Diskret realisierte Sensoranpassschaltung.

A. Die Entwicklungsumgebung und der Fertigungsprozess des Chips HAB10

Das Herzstück der Entwicklung stellt der CMOS-Chip HAB10 dar, auf welchem der SPC bzw. OnChip Temperatursensor, der TDC sowie ein Zähler integriert sind. Der Schaltungsentwurf des HAB10 erfolgt mit der Entwicklungsumgebung Tanner Tools Pro V.15 Design Suite. Im Design Flow befindet sich der Chip aktuell in der Layout-Phase, im Folgenden werden die Simulationsergebnisse präsentiert. Ein Großteil der Hauptkomponenten wird im full-custom Design entwickelt, der Zähler ist aus Standardzellen aufgebaut. Der Halbleiterchip wird in einen 150 nm CMOS Prozess in Kooperation mit EUROPRACTICE gefertigt.

III. Einzelkomponenten des US^{2}C

Im Rahmen dieser Arbeit werden ausschließlich die Sensoranpassschaltung sowie die Signal-zu-Puls Konvertierung (SPC) im Detail beschrieben. Die Entwicklung des zyklischen Pulsschwund TDC ist nicht Gegenstand dieser Arbeit.

A. Die Sensoranpassschaltung

Die analoge Sensoranpassschaltung wurde mit dem Ziel entwickelt, eine möglichst universell einsetzbare Schnittstelle zwischen Sensorik und Halbleiterchip zu realisieren. Bisherige Schwerpunkte sind die Druck- und Temperaturmessung. Zentrales Element stellt der Instrumentenverstärker INA333 [6] (Texas Instruments) dar. Durch externe Beschaltung werden Verstärkung und Offsetspannung so eingestellt, dass Sensorsignale am Eingang dem definierten Steuerspannungsbereich des SPC angepasst werden. Die Auswahl des Sensors erfolgt durch Jumpereinstellung. Zu den Sensorsignalen gehören sieben bereits vorkonfigurierte Sensortypen (Tabelle 1), welche einen breiten Einsatz in industriellen Prozessen finden.

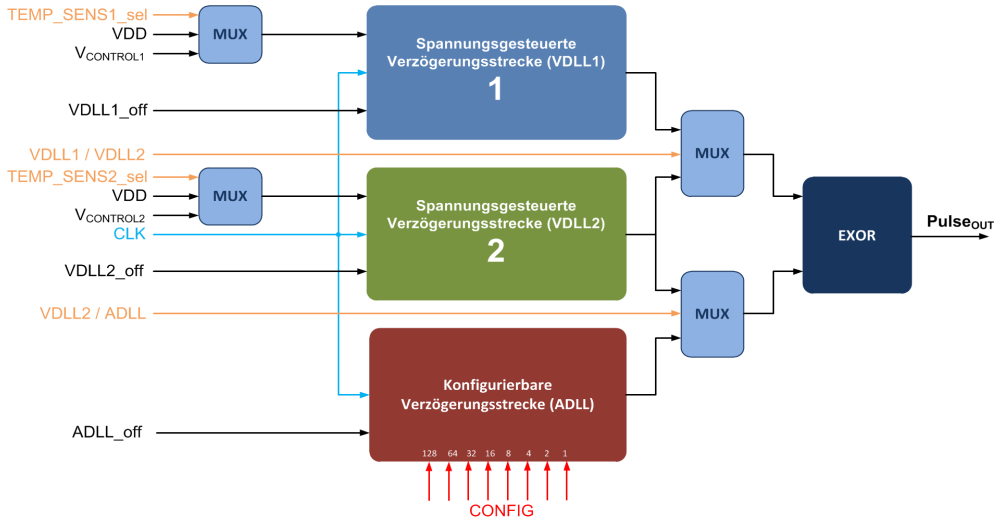


Abbildung 4: Aufbau des konfigurierbaren Signal zu Puls Konverters (SPC).

Tabelle 1: Übersicht der vorkonfigurierten Sensortypen zum Anschluss an die Sensoranpassschaltung.

Sensortyp	Phsyikalische Größe
Metall – Dünnfilm - Sensor	Druck
Keramik – Dickschicht - Sensor	Druck
Piezoresistiver Sensor	Druck
Thermoelement Typ K / L	Temperatur
Pt100 / Pt1000 Widerstands-thermometer	Temperatur

Neben diesen können weitere Sensoren ausgewertet werden. Dazu wird eine externe Sensorschaltung durch Buchsen (siehe Abbildung 3) mit der Anpassung verbunden. Für die Sensorauswertung sind wahlweise eine Konstanspannungs- als auch eine Konstantstromquelle vorgesehen. Die Stabilität des Aufbaus gegen externe Störungen und Temperaturschwankungen wurde durch Messungen in der EMV- sowie in der Temperatur-Kammer bestätigt. Die Anpassschaltung stellt am Ausgang dem nachfolgenden SPC ein Signal mit einer maximalen Spanne von 900 mV und einer minimalen Spanne von 400 mV bei einem Offset zwischen 600 mV und 1,1 V zur Verfügung.

B. Der SPC mit OnChip Temperatursensor

Die Funktion des SPC wird aus drei Konfigurationsmöglichkeiten durch externe Beschaltung eingestellt. Diese sind:

- OnChip Temperaturmessung
- Sensorsignalwandlung
- Differentielle Sensorsignalwandlung

Wie in Abbildung 4 erkennbar besteht der SPC aus zwei identischen spannungsgesteuerten und einer

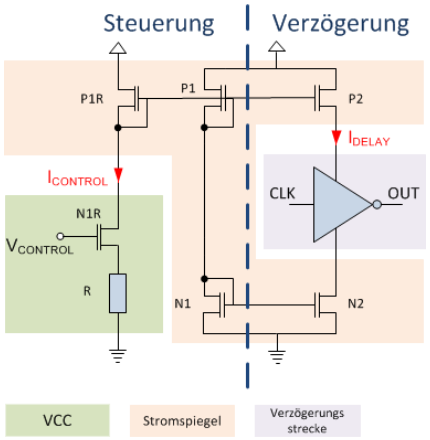


Abbildung 5: Blockweiser Aufbau der spannungsgesteuerten Verzögerungsstrecke.

konfigurierbaren Verzögerungsstrecke [7]. Zur Energieeinsparung ist es möglich, einzelne Verzögerungsstrecken durch Aktivierung der Pins VDLL1_off, VDLL2_off bzw. ADLL_off bei Bedarf abzuschalten. Die Selektionspins TEMP_SENS1_sel bzw. TEMP_SENS2_sel zum Starten der Temperaturmessung an den Multiplexern (links im Bild), werden je nach Konfiguration des Systems gesetzt. Die Wahl der gewünschten Ausgangspulse zur Differenzbildung durch das EXOR-Gatter erfolgt durch Beschaltung der Multiplexer (rechts im Bild). Die Funktionsweisen der spannungsgesteuerten bzw. konfigurierbaren Verzögerungsstrecken werden nachfolgend beschrieben.

1) Die spannungsgesteuerte Verzögerungsstrecke

Das Funktionsprinzip einer spannungsgesteuerten Verzögerungsstrecke (VDLL) ist in Abbildung 5 illustriert [8]. Die von der Sensoranpassung generierte sensorgrößenabhängige Steuerspannung $V_{CONTROL}$ am Eingang des Spannungs-zu-Strom Konverters (VCC),

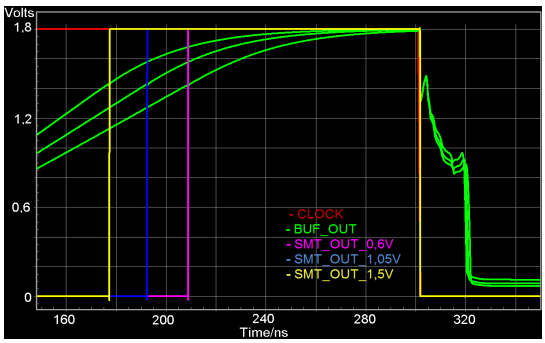


Abbildung 6: Simulationsergebnis des Ausganspulses (SMT_OUT_xxV) der spannungsgesteuerten Verzögerungsstrecke für die Steuerspannungen 0,6 V, 1,05 V und 1,5 V. Das Ausgangssignal der Inverterzelle vor dem Schmitt-Trigger (BUF_OUT) ist grün bzw. das zu verzögernde Taktsignal (CLOCK) rot gekennzeichnet.

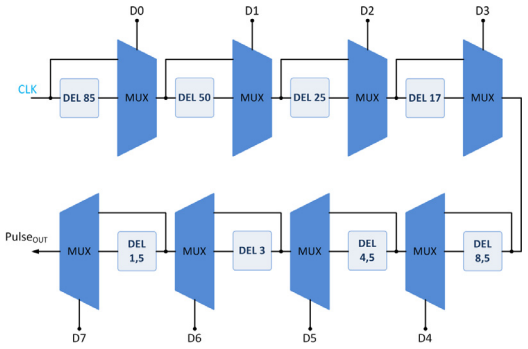


Abbildung 7: Aufbau der konfigurierbaren Verzögerungsstrecke bestehend aus acht Verzögerungszellen und acht Multiplexern. Die Ziffern innerhalb der Verzögerungsblöcke beschreiben deren Verzögerungszeit in ns bei "typical-mean" Bedingungen.

wird in einen linearen Steuerstrom $I_{CONTROL}$ zwischen 5 bis 55 µA konvertiert. Durch die Stromspiegel P1R, P1 und P2 sowie N1 und N2 wird dieser Steuerstrom in den für eine Verzögerung von max. 30 ns benötigten Arbeitsstrom I_{DELAY} von 80 bis 105 µA angehoben. Dieser Strom, der durch das Verzögerungsglied bestehend aus Invertergattern fließt, bestimmt dessen stromabhängige Schaltzeiten t_{pHL} bzw. t_{pLH} [5].

$$t_{pHL} = \frac{C_{load}}{\mu_n C_{OX}(V_{DD} - V_{Tn})} \frac{L}{W}\left[\frac{2V_{Tn}}{V_{DD} - V_{Tn}} + ln\left(\frac{4(V_{DD} - V_{Tn})}{V_{DD}}\right) - 1\right]$$

$$t_{pLH} = \frac{C_{load}}{\mu_p C_{OX}(V_{DD} - |V_{Tp}|)} \frac{L}{W}\left[\frac{2|V_{Tp}|}{V_{DD} - |V_{Tp}|} + ln\left(\frac{4(V_{DD} - |V_{Tp}|)}{V_{DD}}\right) - 1\right]$$

Um die gewünschte Verzögerung zu erzielen, besteht das Verzögerungsglied aus einer Aneinanderreihung zwölf identischer Inverterzellen, wobei jede Inverterzelle aus fünf parallelgeschalteten Invertergattern aufgebaut ist, um die Lade- bzw. Entladekapazität der Inverterzelle zu erhöhen. Für einen definierten Schaltpunkt sorgt ein abschließender Schmitt-Trigger [8], welcher bei Erreichen des oberen Schaltpunktes V_{SPH} = 1,4 V den Ausgangspuls bei steigender Taktflanke generiert. Der untere Schaltpunkt des Schmitt-Triggers V_{SPL} spielt hier keine Rolle, da die fallende Flanke des Ausgangspulses mit fallender Taktflanke erzeugt wird. Die Erzeugung des Ausgangspulses der spannungsgesteuerten Verzögerungsstrecke für drei unterschiedliche Steuerspannungen wird durch das Simulationsergebnis in Abbildung 6 beschrieben. Die Temperaturabhängigkeit der VDLL nimmt mit steigender Steuerspannung ab. Die minimale Temperaturabhängigkeit liegt durch entsprechende Dimensionierung im Punkt $V_{CONTROL} = V_{DD}$, hier ist die Verzögerungszeit über einen Temperaturbereich von -10 bis 80 °C nahezu konstant (Simulationsergebnis Abbildung 11). Diese Eigenschaft der VDLL wird in der Konfiguration "Temperatursensor" zur Referenzmessung verwendet.

2) Die konfigurierbare Verzögerungsstrecke

Die konfigurierbare Verzögerungsstrecke (ADLL) besteht aus acht unterschiedlichen Verzögerungsgliedern und acht Multiplexern [7]. Die Verzögerungsglieder unterscheiden sich in der Anzahl von Invertergattern und damit durch ihre unterschiedlichen Verzögerungszeiten. Durch externe Beschaltung der Multiplexer werden die einzelnen Verzögerungsglieder aktiv bzw. inaktiv geschaltet (Abbildung 7). Die Verzögerungszeit der ADLL berechnet sich somit zu:

$$t_{pADLL} = 8 \times t_{pMUX} + \sum t_{pDEL,AKTIV}$$

Durch die variable Verzögerungszeit ist es möglich, Temperaturabhängigkeiten entgegenzuwirken oder Offsetzeiten einzustellen, wobei unter Vernachlässigung der Verzögerungszeiten der Multiplexer t_{pMUX} die minimale Auflösung 1,5 ns beträgt. Die Verzögerungszeit der konfigurierbaren Verzögerungsstrecke ist stark temperaturabhängig. Dies kommt in der Konfiguration "Temperatursensor" zum Tragen.

3) Konfigurationsmöglichkeiten des SPC

Im Konfigurationsmodus der OnChip Temperaturmessung werden die Taktverzögerungen der konfigurierbaren Verzögerungsstrecke sowie einer der spannungsgesteuerten Verzögerungsstrecken miteinander verglichen. Genutzt wird hier zum Einen die geringe Temperaturabhängigkeit der VDLL bei Steuerspannung $V_{CONTROL} = V_{DD}$ (siehe Simulationsergebnis Abbildung 11). Zum Anderen wird zur Differenzmessung die stark temperaturabhängige Pulsverzögerung der ADLL herangezogen. Der resultierende Ausgangspuls des SPC ist somit temperaturabhängig. Das Ergebnis der Differenzbildung ist unabhängig von der Wahl der VDLL.

Das Gleiche gilt für die VDLL's im Modus der Sensorsignalwandlung. Auch hier wird die Differenz der Pulsdauer einer VDLL und der ADLL ermittelt. Im

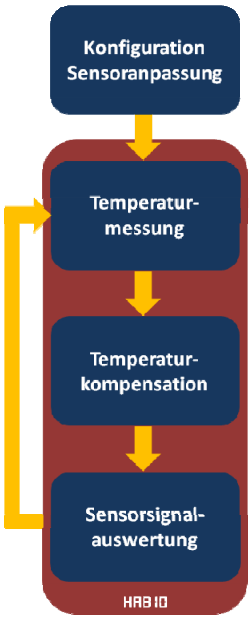


Abbildung 8: Messablauf einer Messung mit dem US^{2}C.

Gegensatz zur Temperaturmessung ist diese Differenz jedoch sensorgrößenabhängig. Am Steuerspannungseingang des SPC liegt nun die von der Sensoranpassung generierte Steuerspannung $V_{CONTROL}$ an. Die Aufgabe der ADLL ist es, die Temperaturabhängigkeit der VDLL zu kompensieren, indem bezugnehmend auf die gemessene OnChip Temperatur eine entsprechende Konfiguration der ADLL vorgenommen und diese somit der Temperatur nachgeführt wird. Zudem kann der Zeitoffset t_{OFFSET} des Ausgangspulses durch Konfiguration den Eigenschaften des nachfolgenden TDCs angepasst werden.

Bei der differentiellen Sensorsignalwandlung spielen nun beide VDLL's eine Rolle, wohingegen die ADLL aus Energiespargründen deaktiviert wird. Die differentielle Sensorsignalauswertung wurde vor allem zur Auswertung von Widerstandsmessbrücken konzipiert. Dazu liegt eine der Steuerspannungen auf einem festen Potenzial während die zweite Steuerspannung mit der Sensorgröße variiert. Mit Hilfe einer vorangehenden OnChip Temperaturmessung wird die Temperaturabhängigkeit der resultierenden Pulsdifferenz controllerseitig kompensiert. Somit ist das Messergebnis auch im Modus der differentiellen Sensorsignalmessung temperaturunabhängig.

C. Die Mikrocontrollerschaltung

Die Mikrocontrollerschaltung deckt die Schnittstellen zum Hochschulchip, zum Anwender sowie zur Messtechnik ab, wobei die Kommunikation mit dem Chip die Hauptaufgabe darstellt. Neben der Konfiguration des SPC's und des TDC's gilt es, den Zählerstand dem aktuellen Modus entsprechend in den korrespondierenden Messwert zu konvertieren. Über die Schnittstelle zum Anwender werden Einstellungen, wie zum Beispiel die Wahl des Sensoreingangs, sowie die Auswertung der Messergebnisse an einem PC ermöglicht. Der Datenaustausch mit der Messtechnik dient der Verifizierung des Systems durch einen automatisierten Messablauf, wobei die Steuerspannungen von einer Digital-zu-Analog Wandlung generiert werden. Die computerseitige Kommunikation mit der Messtechnik zur Datenerfassung wird mittels eines Phyton-Scripts automatisiert.

IV. Der Messablauf

In Abbildung 8 ist der Messablauf des Systems in Form eines Blockschaltbildes illustriert. Im ersten Schritt wird die Konfiguration der diskreten, vom System abgesetzten Sensoranpassschaltung vorgenommen. Entsprechend der Messaufgabe wird hierzu der benötigte Sensoreingang aktiviert. Die Verstärkung und der Spannungsoffset so eingestellt, dass die Ausgangssignale im definierten Steuerspannungsbereich von 0,6 bis 1,5 V liegen. Durch die Mikrocontrollerschaltung kann diese Aufgabe zukünftig ebenfalls von der Anwendersoftware übernommen werden. Im Schritt der OnChip Temperaturmessung arbeitet der SPC durch entsprechende Konfiguration als Temperatursensor und liefert dem nachfolgenden TDC einen temperaturabhängigen Zeitimpuls. Die Temperaturinformation in Form eines Zählerstandes wird daraufhin vom µC verarbeitet und zur Temperaturkompensation herangezogen. Hierbei gilt es, zwei Fälle zu unterscheiden. Arbeitet das System im Schritt der Sensorsignalauswertung im Modus der nicht differentiellen Sensorsignalwandlung, so wird bei der Temperaturkompensation aus einer LookUp-Tabelle der Wert zur Konfiguration der ADLL entnommen und diese entsprechend beschaltet. Im Falle einer differentiellen Sensorsignalwandlung wird entsprechend der gemessenen OnChip Temperatur ein Kompensationsfaktor berechnet und dieser auf das Resultat der Signalauswertung angewendet. Sind die ADLL konfiguriert bzw. der Kompensationsfaktor berechnet, kann die Sensorsignalwandlung durchgeführt werden. Hierzu werden die benötigten Verzögerungsstrecken aktiviert und mit Steuerspannungen gespeist. Ergebnis ist ein sensorgrößenabhängiger Zeitimpuls welcher vom TDC in digitaler Form der nachfolgenden Elektronik zur Verfügung gestellt wird. Für einen Messzyklus werden im Zuge der Simulation 2 ms benötigt, wobei dieser eine Temperaturmessung, sowie vier Sensorauswertungen beinhaltet. Aufgrund einer Messrate von 500 Hz bei der OnChip Temperaturmessung gelten für die nachfolgenden vier Auswertungen die einmalig ermittelten Kompensationswerte. Die Messrate der Sensorsignalauswertung beträgt somit 2 kHz.

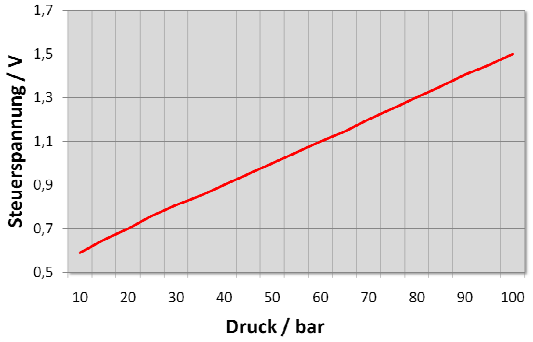


Abbildung 9: Steuerspannung am Ausgang der Sensoranpassschaltung als Funktion des Druckes.

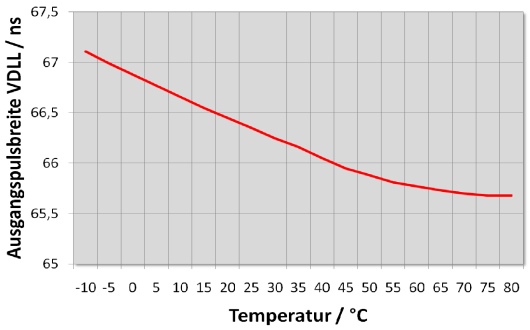


Abbildung 11: Ausgangspulsbreite der temperaturkompensierten spannungsgesteuerten Verzögerungsstrecke als Funktion der Temperatur.

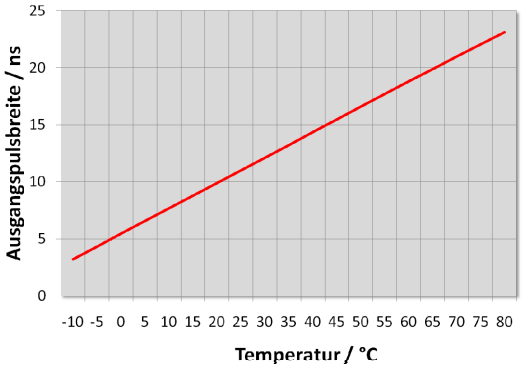


Abbildung 10: Ausgangspulsbreite des Temperatursensors als Funktion der Temperatur.

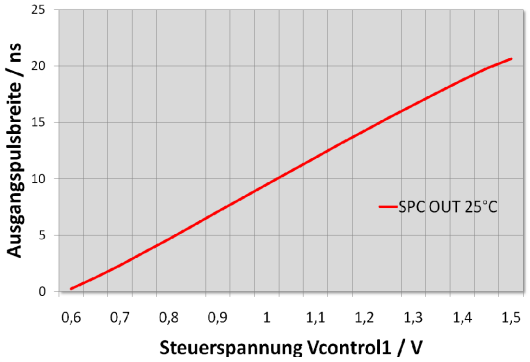


Abbildung 12: Ausgangspulsbreite des SPC bei der nicht differentiellen Signalkonvertierung als Funktion der Steuerspannung bei Raumtemperatur .

V. Ergebnisse

Das gemessene Steuerspannungssignal am Ausgang der Sensoranpassschaltung ist für das Beispiel einer Druckmessung mit einem Metall-Dünnfilm-Sensor im Druckbereich von 10 bis 100 bar in Abbildung 9 dargestellt. Es ist zu erkennen, dass die Sensorsignalspanne von 10 mV in den definierten Steuerspannungsbereich von 0,6 bis 1,5 V angehoben wird. In Abbildung 10 ist die simulierte Ausgangspulsbreite des SPC im Modus der Temperaturmessung über einem Temperaturbereich von -10 bis 80 °C dargestellt. Die Pulsbreitenänderung über den gesamten Temperaturbereich beträgt hier 20 ns. In Kapitel III.B.3) ist beschrieben, dass die VDLL in diesem Modus nur eine geringe Temperaturabhängigkeit aufweist. Abbildung 11 zeigt das zugehörige Simulationsergebnis. Die Ausgangspulsbreite fällt zwischen -10 bis 80 °C nur um 1,4 ns. Abbildung 12 zeigt das zugehörige simulierte Ergebnis der temperaturkompensierten Sensorsignalwandlung bei Raumtemperatur. Für eine Druckmessung im Bereich von 10 bis 100 bar erhalten wir somit am Ausgang des SPC eine simulierte Pulsbreite zwischen 0 bis 21 ns bei variablem Offset. Die Simulationsergebnisse berücksichtigen die vom Halbleiterhersteller definierten Worst-Case Fälle.

VI. Ausblick

Das Konzept eines universellen Sensorsignalkonverters (US^2C) auf Basis der Zeit-zu-digital Wandlung wurde vorgestellt. Die Entwicklung der Sensoranpassschaltung sowie die Simulation des Systems ist abgeschlossen, das Kernbauelement, der CMOS-Chip HAB10, der in 150nm Technologie gefertigt werden wird, befindet sich in der Layoutphase. Zukünftige Arbeiten sind auf die Evaluation des gefertigten Chips sowie auf die Entwicklung einer Auswerteelektronik und -software für einen automatisierten Messablauf gerichtet.

Danksagung

Diese Arbeit wurde im Rahmen des Forschungsschwerpunktes „Intelligente Sensorik“ durchgeführt, den das Bayerische Staatsministerium für Wissenschaft, Forschung und Kunst an der Hochschule für angewandte Wissenschaften, Aschaffenburg, eingerichtet hat. Die Autoren danken dem Unternehmen

WIKA Alexander Wiegand SE & Co. KG für technische Unterstützung.

LITERATURVERZEICHNIS

[1] E. Gassmann, A. Gries, „Elektronische Druckmesstechnik: Grundlagen, Anwendungen und Geräteauswahl“ *[Landsberg/Lech]: Verl. Moderne Industrie*, 2009.

[2] G. Roberts, M. Ali Bakhshian, “A Brief Introduction to Time-to-Digital and Digital-to-Time Converters” *IEEE Transactions on Circuits and Systems II*: Express Briefs, Vol. 57, No. 3, pp. 153–157, 2010.

[3] F. Bernhard, Ed, „Technische Temperaturmessung: Physikalische und meßtechnische Grundlagen, Sensoren und Meßverfahren, Meßfehler und Kalibrierung ; Handbuch für Forschung und Entwicklung, Anwendungenspraxis und Studium“ *Springer*, Berlin 2004.

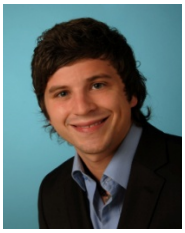
[4] C. Poki, C. C. Chun, C. T. Chin, F. L. Wen, “A time-to-digital-converter-based CMOS smart temperature sensor” *IEEE Journal of Solid State Circuits*, Vol. 40, No. 8, pp. 1642–1648, 2005.

[5] Daniel Fuchs, Ulrich Brunsmann, Andreas Broenner, “An experimental test chip for TDC-based digital sensors” *Workshop der Multiprojekt-Chip-Gruppe Baden-Würtemberg*, Reutlingen, pp. 67 – 78, 2010.

[6] Texas Instruments, “Datenblatt INA333 - Micro-Power (50 uA), Zero-Drift, Rail-to-Rail Out Instrumentation Amplifier”. 2008

[7] C. Poki, C. C. Chun, H. P. Yu, M. W. Kai, S. W. Yu, “A Time-Domain SAR Smart Temperature Sensor With Curvature Compensation and a 3sigma; Inaccuracy of -0.4°C - +0.6°C Over a 0°C to 90°C RangeS” *IEEE Journal of Solid State Circuits*, Vol. 45, No. 3, pp. 600–609, 2010.

[8] R. J. Baker, “CMOS: Circuit Design, Layout, and Simulation” *IEEE Press*, 2nd ed. Hoboken, NJ: Wiley-Interscience [u.a.], 2008.



Daniel Fuchs erhielt den den akademischen Grad des B. Eng in Elektro- und Informationstechnik im Jahr 2010 von der Hochschule Aschaffenburg, wo er sein Masterstudium in Elektro- und Informationstechnik voraussichtlich im Jahr 2011 beendet. Seine Forschungsinteressen liegen im Bereich der integrierten Sensorverarbeitung.

Andreas Brönner erhielt den akademischen Grad des B. Eng. in Elektro- und Informationstechnik im Jahr 2010 von der Hochschule Aschaffenburg, wo er sein Masterstudium in Elektro- und Informationstechnik voraussichtlich im Jahr 2011 beendet.. Seine Forschungsinteressen liegen im Bereich der digitalen Sensorik und Zeit-zu-Digital Wandlung.

Ulrich Brunsmann ist Professor für Elektronische Bauelemente und Computational Intelligence an der Hochschule Aschaffenburg. Zuvor leitete er die Sensor- und Elektronikentwicklung für die Weltraumastronomie im Battelle-Institut, Frankfurt. Er arbeitete an der Universität Gießen auf dem Gebiet der Festkörperoberflächenphysik und erhielt dort 1977 den Grad des Dr. rer. nat. und 1972 den Grad des Diplom-Physikers.

Hochschule Offenburg
University of Applied Sciences

Entwicklung eines 16/32-bit Prozessorkerns für einen PDA mit JTAG-Schnittstelle und Implementierung in einer 0,18µm-CMOS Technologie

Benjamin Dusch, Sebastian Stickel, Andreas Kreker, Dirk Jansen

Zusammenfassung—Der 16/32-bit Prozessorkern SIRIUS-JANUS, entwickelt an der Hochschule Offenburg, wird mit Peripherieeinheiten wie SPI-Controller, Audioschnittstelle, drei Timern, Interrupt-Controller, Display-Controller, Controller für externen SRAM, PLL-Controller, ROM, alle ebenfalls entwickelt an der Hochschule Offenburg, und RAM als ASIC-Design „PDA-ASIC VERSION_2“ entwickelt und in 0,18 µm-Zieltechnologie implementiert. Der neu entwickelte Chip ist eine Folgeversion des bereits 2009 in Silizium gefertigten und erfolgreich erprobten ASIC-Designs „PDA-ASIC“. Das aus diesem Jahr stammende, in weiten Teilen übernommene Gesamtdesign wird hinsichtlich der Leistungsdaten wesentlich verbessert, wobei die Multiplikation beschleunigt und die Taktrate des Systems durch Syntheseoptimierung gesteigert wird. Das Design wird durch Teststrukturen wie Boundary Scan (JTAG), Scan und Test Points ergänzt, um automatische Testbarkeit sowie Debugging des gefertigten ASICs im Sinne von Design For Testability (DFT) zu ermöglichen. Der Design-Flow vom Quellkode über Syntheseoptimierung bis hin zum Layout wird skizziert und der Entwicklungsprozess von der Logik bis zum gefertigten ASIC-Design verdeutlicht.

***Schlüsselwörter*—Prozessorkern, Prozessor, PDA, ASIC, Chip, JTAG, Boundary Scan, Scan Chain, Test Points, DFT, Design Flow**

I. Einleitung

Vor einigen Jahren wurde an der Hochschule Offenburg im Institut für Angewandte Forschung (IAF) die Entwicklung einer eigenen Prozessorkernfamilie für Lehre und Forschung begonnen. Das erste Mitglied dieser Familie ist der 16/32-bit-Prozessorkern SIRIUS-JANUS [1], geeignet sowohl in der 16-bit-Rolle wie auch als 32-bit-Kern.

Benjamin Dusch, Benjamin.Dusch@HS-Offenburg.de, ASIC Design-Center, Hochschule für Technik, Wirtschaft und Medien – Offenburg, University of Applied Sciences, Badstraße 24, 77652 Offenburg

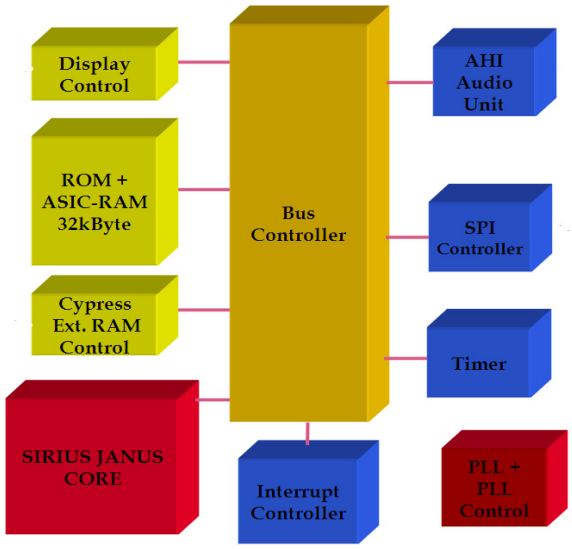


Abbildung 1: Prozessorkern SIRIUS-JANUS mit verbauter Peripherie des ASIC-Designs "PDA-ASIC" (Version 2009)

Weitere Familienmitglieder sind der SIRIUS-TINY, ein reiner 16-bit-Prozessor für kleine Steuerungsaufgaben und der SIRIUS-HULK, ein reiner 32-bit-Bolide mit Cache und Harvard-Architektur. Diese Entwicklungen stehen unter dem Aspekt der Lehre und Demonstration der heutigen technischen Möglichkeiten und dienen der Vertiefung der Ausbildung der Studenten, insbesondere im Masterbereich.

Der SIRIUS–JANUS wurde bereits seit 2007 in 0,35 µm CMOS ASICs [2] integriert, wo er mit einer Taktfrequenz bis etwa 80 MHz arbeitet. Mit dem Übergang auf die UMC 0.18 µm Technologie [3] und der damit verbundenen Möglichkeit, die aktuell vorhandenen Entwurfsprogramme auszureizen, sollte in einem Re-Design des bereits 2009 ebenfalls in UMC 0.18 µm erfolgreich gefertigten „PDA-ASIC“ [4] die volle Leistungsfähigkeit der Technologie und der Entwurfswerkzeuge untersucht werden. Weiterhin sollte eine Erweiterung um eine JTAG-Teststruktur erfolgen, um automatische Tests wie auch Debugging ermöglichen zu können.

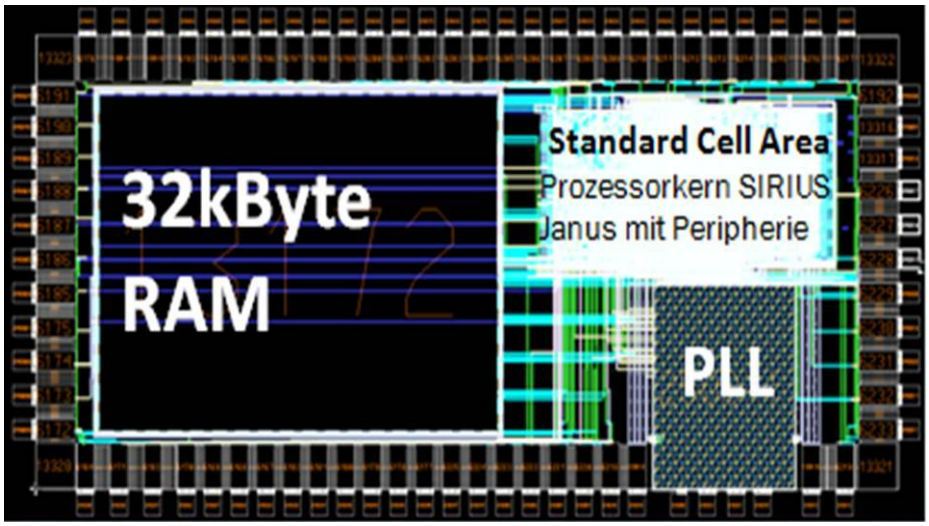


Abbildung 2: Layout des ASICs "PDA-ASIC" aus dem Jahre 2009

Tabelle 1: Eigenschaften des SIRIUS-JANUS (Version 2009)

Performance	SIRIUS-JANUS
Internal data bus	16/32 Bit
Address space	4GByte
OS feasible	BIOS-RTOS
Multiplier	32 x 32
Instruction set	58
High level Language	C-Compiler
Cache-System	No
Verified on ASIC	Yes
MIPS average / MHz	0.8

Die im Design „PDA-ASIC“ 2009 in 0,18 µm verbauten Elemente der Peripherie sind in Abbildung 1 dargestellt. Verbaut wurde neben dem Prozessorkern SIRIUS-JANUS ein 32 kByte RAM (IP von UMC), ein ROM, eine Displaysteuerung, eine Steuerung für einen extern anzuschließenden RAM, eine AHI-Audioeinheit, ein SPI-Controller, drei Timer und eine PLL (IP von UMC) mit Steuerlogik.

In der Tabelle 1 sind die Kerndaten des Prozessorkerns aus dem Jahre 2009 aufgezählt. Die maximal mögliche Taktfrequenz ist nicht aufgeführt, da diese von der Zieltechnologie abhängig ist. In 0,35 µm Technologie AMI sind das etwa 80 MHz, im „PDA-ASIC“ in 0.18 µm UMC sind das gemessen etwa 160 MHz. Eine weitere Leistungssteigerung erscheint möglich, da die Optimierungsmöglichkeiten in der Synthese bisher noch nicht ausgeschöpft wurden. Das „PDA-ASIC VERSION_2“ sollte zudem industriellen Prüfverfahren entsprechen und DFT Strukturen enthalten. Hier wurden bereits erste Erfahrungen in einem etwas einfacheren Design, dem Front-End-ASIC „DQPSK“ gewonnen, ein Design der hier angestrebten Komplexität stellt jedoch eine Herausforderung dar.

Mit der Neuauflage des PDA-ASICs sollte gleichzeitig die Performance bei möglichst geringerer Leistungsaufnahme pro MHz gesteigert werden, weshalb das Optimierungspotential zunächst analysiert wurde. Damit wurden zum ersten Mal auch die Werkzeuge zur Leistungsanalyse in Betrieb genommen, da die UMC-Bibliothek alle hierfür erforderlichen Daten enthält.

Die Analyse des bestehenden Designs im Programm Cadence Encounter ergab, dass das Design „PDA-ASIC“ bezüglich der maximalen Taktrate nicht ausgereizt ist und nur einen Systemtakt von maximal etwa 100 MHz (Synthese) ermöglicht. Die im Rechenwerk integrierte Multiplikation 32 x 32 verwendet zur Beschleunigung eine Pipelinestufe, was in der aktuell verwendeten 0,18 µm Technologie nicht mehr notwendig ist. Der zusätzliche Taktschritt für die Pipelinestufe war im Zusammenhang mit der doch deutlich langsameren 0,35 µm Technologie notwendig gewesen.

Abbildung 2 zeigt das Layout des „PDA-ASIC“. Der Chip ist padbegrenzt, die verfügbare Fläche von 4,10 mm2 ist mit 3,26 mm^2 wenig effektiv genutzt. 20,6% der zur Verfügung stehenden Fläche bleiben noch frei. Die im „PDA-ASIC“ genutzte Routing-Fläche zwischen dem Floorplan der Standardzellen und den IPs wird in dieser Rechnung nicht berücksichtigt, da das Routing der Makros und des Standardzellen-Floorplans nach außen hin im Vergleich zum Placing unbedeutend wenig Fläche benötigt.

Es bestehen damit noch Möglichkeiten, das System geschwindigkeits- als auch flächenmäßig zu optimieren. Drei wesentliche Punkte rechtfertigten nun also das Re-Design des ASICs:

- die Einführung von DFT,
- die Optimierung der Rechengeschwindigkeit
- und die effizientere Ausnutzung der Fläche.

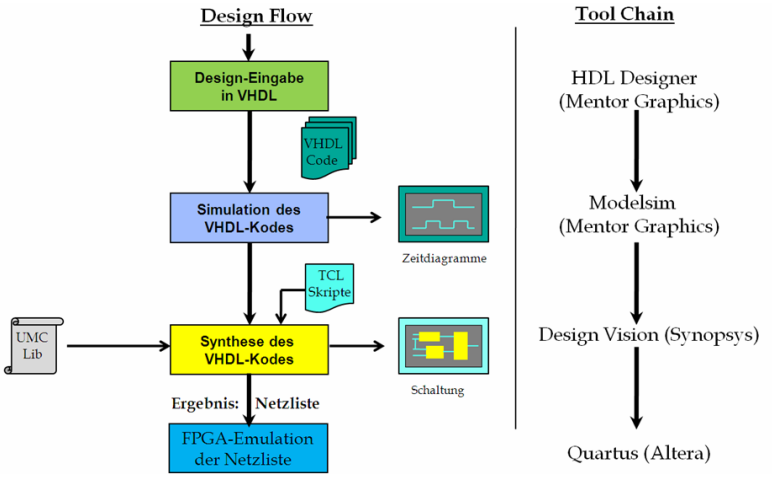


Abbildung 3: IC-Flow - vom Quellkode zur Netzliste

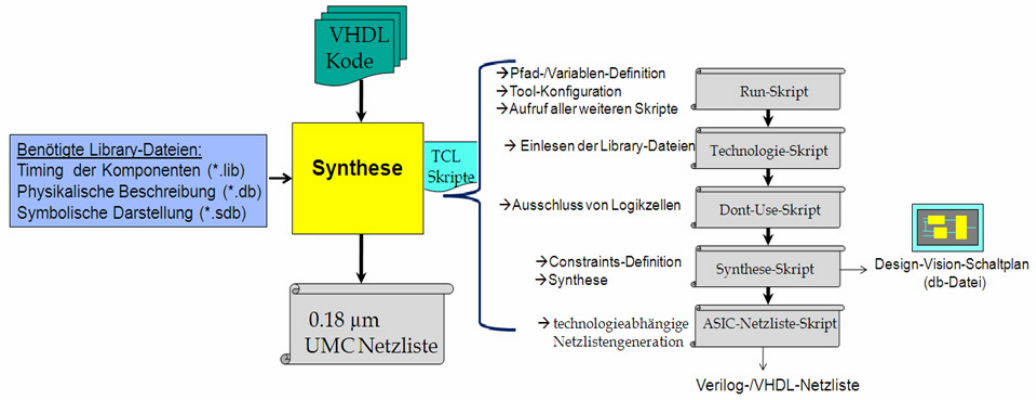


Abbildung 4: Flow der per Skript automatisch ablaufenden Synthese

II. IC-Flow

Basis des richtigen Verständnisses für die vorgenommenen Optimierungsvorgänge ist der im Folgenden erläuterte IC-Flow, welcher alle Vorgänge umfasst, die durchlaufen werden müssen, um aus Quellkode ein zu fertigendes Layout generieren zu können. Aspekte des Flows, welche zwecks Optimierung des Designs besonderer Beachtung bedürfen, werden im Folgenden benannt.

Abbildung 3 zeigt den ersten Teil des Ablaufs, welcher bis zur fertigen Netzliste führt. Im „HDL-Designer“ von Mentor wird das Design des ASICs in VHDL beschrieben, in MODELSIM im Zeitbereich auf korrekte Funktionsweise kontrolliert und im Synthesetool „Design Vision“ von Synopsys die Netzliste als Ergebnis der Umsetzung des programmierten Quellkodes in eine direkt hardwarebasierte Schaltung in Zieltechnologie generiert.

Die Synthese erfolgt automatisiert und skriptgesteuert (Abbildung 4). Die Syntheseskripte in der Sprache Tcl sind auf das Design angepasst. Aus dem ersten Skript (Run-Skript) heraus werden das Synthesetool konfiguriert und Verzeichnispfade und Variablen definiert, welche für die automatische Abarbeitung der Synthese benötigt werden. Aus dem ersten Skript werden die weiteren Skripte nachfolgend aufgerufen. So folgen das Einlesen von Bibliotheken der Zieltechnologie (Symbole für Schaltplandarstellung, Zellbeschreibung, Funktion, Timing…) im Technologie-Skript und Definitionen der für die Synthese aus der Bibliothek nicht zu verwendenden Zellen im Don`t-Use-Skript. Das nächste aufgerufene Skript (Synthesize-Skript) umfasst die eigentliche Transformation in eine Schaltung, hier ist der Ansatzpunkt zur Geschwindigkeitsoptimierung zu finden. Rahmenbedingungen (Constraints) werden definiert, die als Zielvorgabe der zu erzeugenden Schaltung dienen, insbesondere das zu erreichende Timing-Ziel des Designs wird umfassend analysiert und definiert. Auch bezüglich der maximalen dynamischen Leistung des Designs werden Constraints eingegeben.

Über inkrementale Synthesedurchgänge bei Verwendung des leistungsstärksten Synthese-Algorithmus des Syntheseprogramms von Synopsys [5] wird die Netzliste des Designs erzeugt und maximal optimiert, solange bis die fertige Schaltung den Zielvorgaben

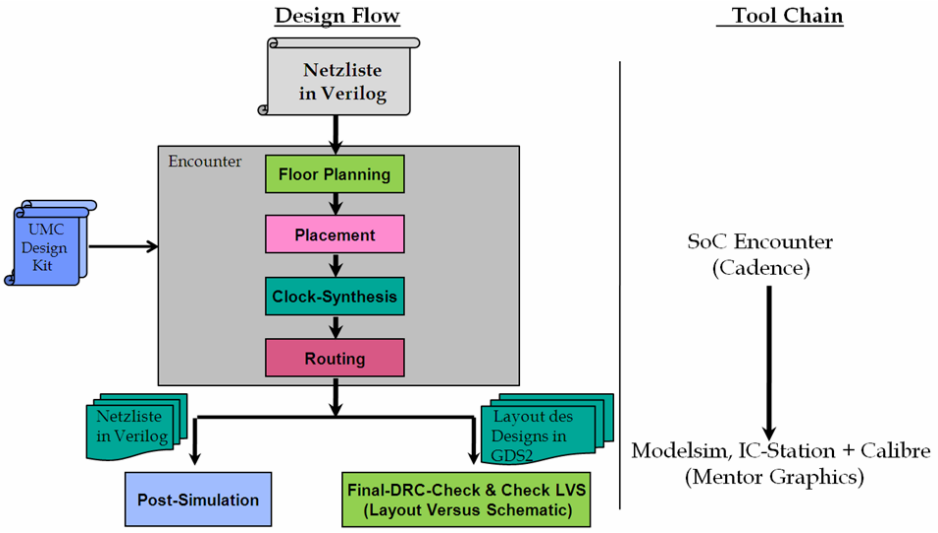


Abbildung 5: IC-Flow - von der Netzliste über das Layout zur Ergebnis-Verifikation

bestmöglich genügt. Ein fertiger Schaltplan kann letztlich nach Vollendung der Synthese für das Design in Zieltechnologie dargestellt werden. Eine Netzliste des fertig synthetisierten Designs wird im letzten Teil des Synthesekomplexes zieltechnologiespezifisch per Skript in Sprachen wie Verilog/VHDL herausgeschrieben.

Die Emulation, also die Nachbildung der durch die Synthese generierten Schaltung im FPGA, bildet den Schluss dieses ersten Teils des Flows und stellt sicher, dass die synthetisierte Schaltung hardwarenahe funktioniert. Hierbei wird die generierte Netzliste wieder in das Entwurfswerkzeug von z.B. ALTERA Quartus II zurückgelesen, wobei die Originalgatter der Bibliothek durch Primitives ersetzt sind. Damit kann die Netzliste verifiziert werden. Das Verfahren greift dabei nicht auf ALTERA-Makros wie z.B. Multiplizierer zurück, sondern bildet die Netzliste direkt ab.

Abbildung 5 zeigt den letzten Teil des Design-Flows. Die generierte Netzliste wird im Layout-Tool „Encounter“ von Cadence als Basis des zu fertigenden Layouts verwendet. Eine Einteilung der verfügbaren Chipfläche in Flächen für Kernkomponenten (Kernlogik des ASICs) und in Fläche für Pad Zellen, stellt den Anfang des Layoutens dar, der „Floorplan“ wird erstellt. Das „Placement“ ist jener Teil des Flows, in dem sämtliche Zellen des Designs platziert werden. Sind die IO- und Power-/Ground-Pad Zellen platziert, folgt die Platzierung der Makros wie RAM oder PLL, anschließend werden die Standardzellen der Kernlogik platziert. Die Clock-Synthese baut anschließend den Clock Tree auf. Dieser Vorgang hat eine gleichmäßige und zeitlich ausbalancierte Taktversorgung aller sequentiellen Zellen zur Folge. Beim „Routing“, dem letzten Part des Layoutens in Cadence Encounter, werden die Verbindungen aller platzierten Zellen untereinander hergestellt. Standardzellen, Pad-Zellen, RAM- und PLL-Makros werden entsprechend der Definition in der Netzliste bei Verwendung mehrerer Metallschichten, in unserem verwendeten Prozess von UMC sind das sechs Metal-Layer, physikalisch verbunden.

Als Ergebnis des Layoutens ergibt sich eine Datei im GDSII-Format, welche den Kern des ASIC-Designs noch ohne Bond Pads beschreibt. Bond Pads werden aber benötigt, um Ein-/Ausgänge und Power-/Groundanschlüsse des ASIC-Designs innerhalb des zu verwendenden Gehäuses mit den nach außen führenden Gehäuse-Pins zu verbinden. Deshalb erfolgt ein Import der Datei im GDS 2-Format in das Programm „IC Station“ von Mentor, wobei automatisch an jede der Pad Zellen ein geeignetes Bond Pad angebracht wird. Die Routine zum automatischen Anbringen der Bond Pads wurde im Rahmen dieser Arbeit erarbeitet, sollen an dieser Stelle aber nicht näher erläutert werden. Auch der finale Design Rule Check (DRC), also die Prüfung auf Fehler im Layout des Designs, welche Regeln des Herstellungsprozesses des ASICs verletzen, und der Layout Versus Schematic (LVS) Check, der Vergleich auf Übereinstimmung von Schaltungsvorgabe und Layout-Ergebnis, werden noch in der „IC Station“ vorgenommen.

Vom fertigen Layout wird in „Encounter“ eine Netzliste in Verilog (wie schon zuvor am Ende der Synthese) herausgeschrieben, welche zur Post-Simulation mit „MODELSIM“ dient. Wie schon zu Beginn des IC-Flows, als die korrekte Funktionalität des Quellkodes im Zeitbereich per Simulation in „MODELSIM“ analysiert wurde, geschieht dies an dieser Stelle nun ein weiteres Mal. Die gewünschte Funktion des ASICs wird über „Forcen“ der ASIC-Eingänge mit definierten Signalen bei Kontrolle der Signale an den ASIC-Ausgängen zwischen dem Prozessorkern und den Peripherieeinheiten verifiziert.

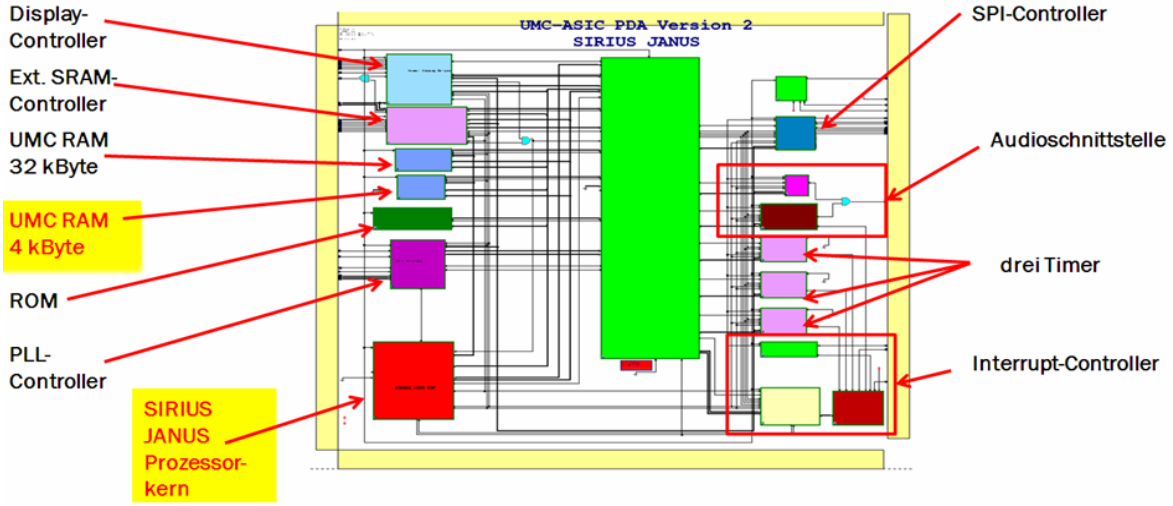


Abbildung 6: Ansicht der Kernlogik des ASIC-Designs "PDA-ASIC VERSION_2" im „HDL-Designer“ von Mentor

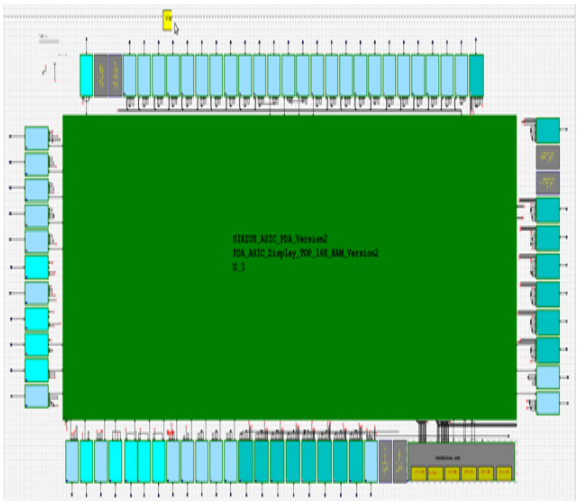


Abbildung 7: Top-Ansicht des ASIC-Designs "PDA-ASIC VERSION_2" im „HDL-Designer“ von Mentor

III. Optimierung des „PDA-ASIC“

Der Beschreibung des Flows, welcher zur Optimierung durchlaufen werden muss, folgt nun die Beschreibung der Optimierungsvorgänge. Das geöffnete Design des ASICs im „HDL-Designer“ von Mentor, Kernlogik in grün, IO-Pad Zellen in blau, Power-/Ground-Pad Zellen in grau, ist im Abbildung 7 zu sehen. Wird eine Hierarchieebene herab auf die Ebene der Kernlogik gestiegen, zeigt sich Abbildung 6, auf welcher der Prozessorkern SIRIUS-JANUS inklusive der Peripherieelemente zu finden ist.

Die Geschwindigkeitsoptimierung des vorhergehenden PDA-ASICs auf maximale Taktrate, inkrementale Synthesedurchgänge und die damit verbundene Designvergrößerung führen noch zu keiner konsequenten Ausnutzung der freien Die-Fläche, sodass ein weiterer

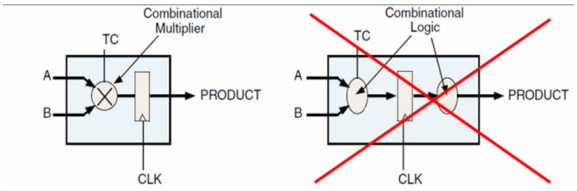


Abbildung 8: Überarbeitung der Logik der ALU: Entfernen der Pipelinestufe im Multiplizierer [5]

RAM (4 kByte) zusätzlich zum vorhandenen RAM von 32 kByte implementiert werden kann.

Die anderen Peripherieeinheiten werden großen Teils unverändert in das neue Design übernommen, kleinere Änderungen, Modifikationen sind für diese Arbeit irrelevant und werden hier nicht erwähnt. Erhöht wird unter anderem die Anzahl der implementierten Timer auf drei. Im Rahmen der Inbetriebnahme des ersten PDA-ASICs hat sich erwiesen, dass dies sinnvoll ist.

Die Optimierung der Logik der ALU folgt als nächster wesentlicher Punkt. Der Prozessorkern SIRIUS-JANUS, dargestellt im „HDL-Designer“ von Mentor (Abbildung 6), beinhaltet alle wesentlichen Komponenten wie Leitwerk, Rechenwerk (ALU), Registersatz, Flagregister, eine Einheit zur Kontrolle der Adressierung und Steuereinheiten für Datenbuseingang und Datenbusausgang. Die ALU wird im neuen Design rein kombinatorisch umgesetzt. Der bisher getaktete zweistufige Multiplizierer (two-stage pipelined multiplier) wird ersetzt durch einen Multiplizierer ohne sequentielle Zellen (Abbildung 8). Eine Taktung der ALU ist somit nicht mehr erforderlich. Die Analyseergebnisse, die zu dieser grundlegenden Änderung des Multiplizierers der ALU geführt haben, sind in Tabelle 2 dargestellt.

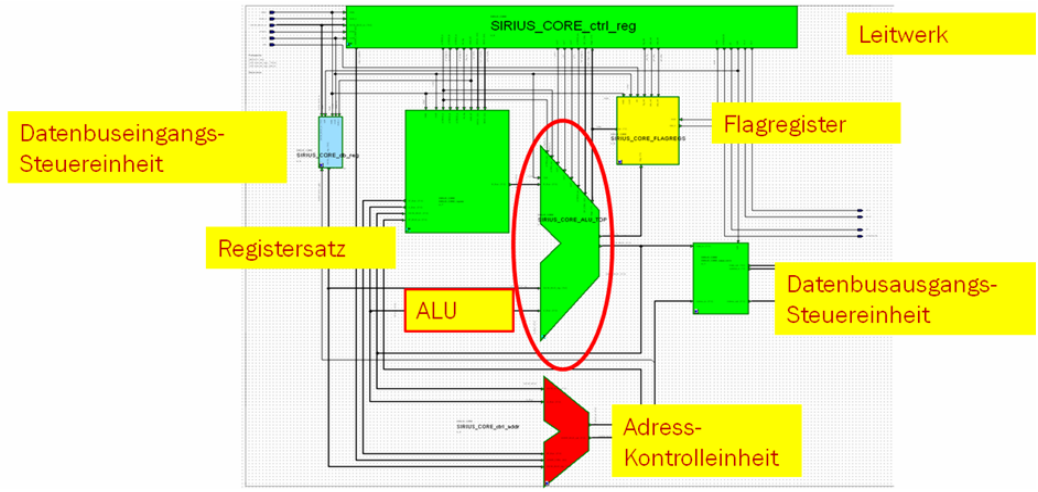


Abbildung 9: TOP-Ansicht des Prozessorkerns SIRIUS-JANUS im „HDL-Designer“ von Mentor

Tabelle 2: Ergebnisse der Modifizierung der ALU – Vergleich vorher/nachher

	Design	*PDA-ASIC-Version_2*	
		ALU mit Multiplizierer pipelined	***ALU mit Multiplizierer kombinatorisch***
ALU allein	register to register delay time	1,76 ns	-
	worst input/output delay time	1,24 ns	-
	delay time:	-	2,62 ns
SIRIUS JANUS Prozessorkern (also inkl. ALU)	register to register delay time	3,51 ns	4,63 ns
	worst input/output delay time	2,12 ns	4,87 ns
	max. Taktrate	**1/3,51 ns = 285 MHz**	**1/4,87ns = 205 MHz**

Geringere max. Taktrate, aber Multiplikation nun in einem statt in zwei Takten!
Prozessor arbeitet somit bei der Multiplikation trotzdem schneller als pipelined

Die Einführung einer Pipelinestufe (Optimierungsversion 1) bringt eine Erhöhung der Taktrate, da der Multiplizierer nun mit 1,76 nsec arbeiten könnte. Demgegenüber bringt die kombinatorische Version (Optimierungsversion 2) in UMC 0.18 µm ein Delay von 2,62 nsec.

Nimmt man aber die weiteren erforderlichen logischen Strukturen in der ALU hinzu, ist der Unterschied nicht mehr so drastisch, es stehen hier 285 MHz gegen 208 MHz (synthetisiert). Da beim Pipeline-Konzept aber ein Takt mehr benötigt wird, der Multiplikationsbefehl damit also zwei Takte an Stelle eines Takts benötigt, ist die Gesamtlaufzeit für die Pipeline-Version damit deutlich ungünstiger. Das soll hier noch genauer beleuchtet werden.

Der Prozessorkern für sich benötigt insgesamt maximal 3,51 nsec Rechenzeit von Register zu Register. Verzögerungszeiten an Ein- oder Ausgängen des Prozessorkerns (benötigte Zeit zum Durchlauf der Kombinatorik von einem Design-Eingang zum ersten Flip Flop-Eingang bzw. benötigte Zeit vom Ausgang des letzten Flip flops eines Pfades durch die Kombinatorik zu einem Design-Ausgang) sind im Vergleich zu den Register-zu-Register-Verzögerungszeiten vernachlässigbar klein und bedürfen keiner Beachtung. Ist der Multiplizierer also zweistufig, ist eine maximale Taktrate des Prozessors von 285 MHz möglich, allerdings werden zwei Takte für Multipliziervorgänge benötigt.

Bei Realisierung einer rein kombinatorischen ALU wird in der ALU bei Verwendung des Multiplizierbefehls eine maximale Verzögerungszeit vom Eingang zum Ausgang der ALU von 2,62 nsec erzielt, was eine knapp 49%ige Verlängerung bedeutet. Die gesamte Rechenzeit des Prozessorkerns betrachtend, reduziert sich der Einfluss der nun rein kombinatorischen, rechenzeitmäßig aufwändigeren ALU auf eine nur noch knapp 39%ige Vergrößerung der Rechenzeit des Prozessorkerns. Die maximale Verzögerungszeit im Kern, nun gemessen an den Ein-/Ausgängen des Prozessors, beträgt 4,87 nsec, was eine maximale Taktrate von 205 MHz ergibt.

Tabelle 3: Ergebnisse des optimierten Prozessorkerns SIRIUS-JANUS - Vergleich vorher/nachher

	Design		PDA-ASIC-Version_2	
			ALU mit Multiplizierer pipelined	ALU mit Multiplizierer kombinatorisch
SIRIUS JANUS Prozessorkern (also inkl. ALU)	max. Taktrate		1/3,62 ns = 276 MHz	1/4,87ns = 205 MHz
	Fläche		274177 µm^2	251375 µm^2
	Zellen	sequentiell	400	368
		kombinator.	10924	10886
	max. dynamische Leistung		50,12 mW	13,40 mW
	Leistung pro MHz		176 µW	65 µW

El. Leistung sinkt enorm durch rein kombinatorische ALU!

Der Prozessorkern ist im Falle der kombinatorischen ALU jetzt bezüglich der maximal möglichen Taktrate gegenüber Optimierungsversion 1 zwar um 39% langsamer geworden, die Rechengeschwindigkeit bei der Multiplikation ist trotzdem vergrößert, da nun in einem statt in zwei Takten multipliziert wird. Da der Prozessorkern in einer Umgebung arbeiten soll (PDA), in welcher sehr häufig multipliziert wird (z.B. JPEG-Verarbeitung), ist der Kern mit rein kombinatorischer ALU damit leistungsfähiger, zudem deutlich kompakter und weniger leistungshungrig wegen Wegfall der zahlreichen Pipelineregister.

Hinsichtlich der benötigten Fläche für den Prozessorkern ergeben sich in beiden Optimierungsversionen keine wesentlichen Unterschiede (Tabelle 3). Der Kern mit kombinatorischer ALU benötigt 8% weniger Die-Fläche. Dies erklärt sich durch die Reduktion der Anzahl der sequentiellen Zellen um 32 (32-Bit-Multiplizierer) bei ungefähr gleichbleibender Anzahl kombinatorischer Zellen. Ein weiterer signifikanter Vorteil der nun taktlosen ALU zeigt sich bei der Leistungsanalyse der Schaltung. Die dynamische Leistung des kombinatorischen Prozessorkerns liegt signifikant unter der Version mit Pipelineregistern. Gegenüber Optimierungsversion 1 sinkt die Leistung auf 65 µW/MHz von zuvor 176 µW/MHz. Die Entscheidung für die Optimierungsversion 2 ist deshalb auch von der elektrischen Leistungsaufnahme her eindeutig.

Einer syntheseseitigen Optimierung des Prozessorkerns SIRIUS-JANUS folgt die syntheseseitige Optimierung aller Peripherieeinheiten des ASIC-Designs, dann eine gesamtoptimierende Synthese des ganzen ASIC-Designs mit dem Ziel einer maximal möglichen Taktrate (siehe IC-Flow in Kapitel II). Die Analyse der Ergebnisse dieser Optimierung im Vergleich zu den Syntheseergebnissen des ersten ASIC-Designs „PDA-ASIC“ folgt in Kapitel V.

IV. Einführen Von DFT

Der gefertigte ASIC soll über eine Testlogik verfügen, die ein automatisches Testen und Debuggen ermöglicht (DFT = Design For Testability). Hierzu werden Schaltungen automatisch hinzugefügt unter Nutzung der Synopsys Tool Chain „Boundary Scan Insertion, Scan Insertion, Test Point Insertion“ [6][7]. Das Debuggen eines ASICs während des Betriebs soll ebenfalls möglich gemacht werden, da es sinnvoll ist, von extern sämtliche Registerinhalte (Zustände der Flip flops) während des Betriebs im Einzeltaktbetrieb auslesen und frei manipulieren zu können. Bei ASIC-Designs, die über interne RAMs verfügen, soll auch der RAM von extern kontrollier- und auslesbar sein. Sowohl RAM-Eingänge als auch RAM-Ausgänge müssen erreichbar sein, um Daten abspeichern und auslesen zu können. Diese Debugging-Funktionen bezüglich des RAMs können nach erfolgter Fertigung des ASICs auch verwendet werden, um einen internen RAM durch Nutzung der Testlogik zu beschreiben und auszulesen.

In vorliegendem Fall soll die Testlogik eine Boundary Scan-Architektur sein, wobei es sich bei der Boundary Scan-Architektur eher um eine Teststeuerungslogik handelt, durch welche andere integrierte Testlogik kontrolliert wird. Im ASIC-Design „PDA-ASIC VERSION_2“ wird die Boundary Scan-Architektur mit mehreren integrierten Scan Chains, eine Scan Chain für die Kernlogik und weitere Scan Chains für die internen ASIC-RAMs, eingeführt.

Dem Design wird ein Test Access Port (TAP, auch als JTAG-Schnittstelle bezeichnet, Abbildung 11) hinzugefügt, durch welchen die Teststeuerungslogik durch Kommandos kontrolliert und ausgelesen wird. Hierzu gehören der Test Mode Select (TMS) zur Steuerung des ebenfalls implementierten Test Access Port Controller (TAP Controller), eine Test Clock (TCK) zur Taktung des ASICs bei Verwendung von Boundary Scan in einer Test-/Debugging-Umgebung,

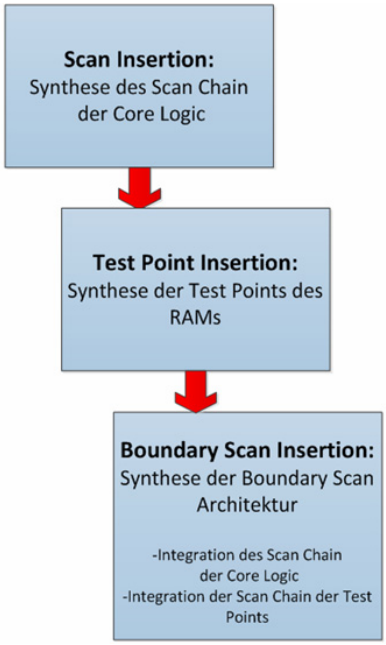


Abbildung 10: Flow der Einführung von DFT in das ASIC-Design des „PDA-ASIC VERSION_2“ [7]

ein Test Data In (TDI) als Eingang für Steuerungs-/Test-Daten und ein Test Data Out (TDO) als Ausgang von Steuerungs-/Test-Daten [8]. In die zentrale Recheneinheit, den Prozessorkern SIRIUS-JANUS, in den ROM sowie in die 36 kByte RAM, in den Buscontroller, in die Displaysteuerung und in die diverse Peripherieeinheiten wie Audioschnittstelle (AHI + Sigma-Delta Wandler), SPI (Serial Peripheral Interface), Timer und in den Interrupt Controller sollen, soweit Flip flops in den verschiedenen Einheiten verbaut sind, Scan Chains implementiert werden. Alle implementierten Scan Chains werden am Ende der Scan Insertion zu einem langen Scan Chain zusammengefasst. Hiernach werden alle RAM-Ein-/Ausgänge mit speziellen Schaltungselementen, Test-Points, ausgestattet, durch welche die Funktionen der Beschreibbarkeit und Auslesbarkeit der RAMs (Debugging-Funktion) ermöglicht werden.

Nach erfolgter Einführung von Scan Chain sowie Observe/Control Test-Points wird das ASIC-Design mit Boundary Scan/JTAG ausgestattet. Die Boundary Scan Insertion umfasst die Integration der Scan Chain als auch die Integration der Test-Points (Abbildung 10). Die Einführung von DFT in das Design erfolgt nach einmaliger Design-Anpassung im Rahmen der Synthese automatisiert und skriptgesteuert. Alle Skripte, die für die Scan/Test Point/Boundary Scan Insertion ausgeführt werden müssen, werden aus Run-Skripten (siehe Kapitel II) heraus automatisch aufgerufen. Der Aufwand, der zur Implementierung von DFT in das ASIC-Design nötig ist, wird in der Tabelle 4 dargestellt. Es ist festzustellen, dass der aufzubringende Flächenmehraufwand sich bei diesem

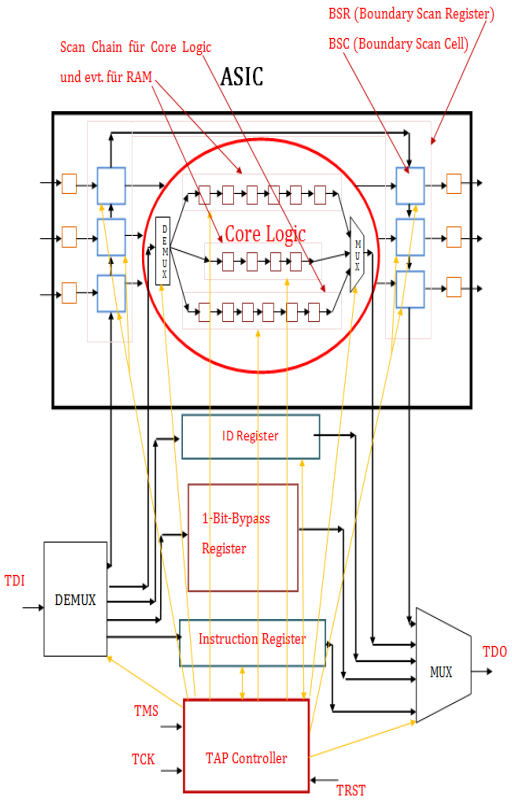


Abbildung 11: DFT-Strukturen: Boundary Scan Architektur mit integrierten mehreren Scan Chains für Kernlogik des ASICs und RAM-Debugging [6]

Tabelle 4: Aufwandsanalyse der Einführung von DFT in das ASIC-Design des „PDA-ASIC VERSION_2“

Design	PDA-ASIC-Version2		
	ohne DFT (kein Scan, keine Test Points, kein Boundary Scan)		*mit DFT (Scan, Test Points, Boundary Scan)*
Fläche A [um²]	1209696		1277270
Anzahl Logikzellen	14544	*nur +* 5,6 %	16998
Anzahl Netze	15014	*nur +* 16,9 %	17643

komplexen Design in Grenzen hält. Es werden zwar 16,9 % mehr Logikzellen implementiert, jedoch lediglich 5,6 % mehr Fläche benötigt. Dieser geringe Flächenzuwachs ist angesichts der enormen Mächtigkeit der neu erlangten Debugging- und Testfunktionalitäten durch DFT gut zu vertreten.

V. Synthese-Ergebnisse

Das neue Design des “PDA-ASIC Version_2” ist nach der syntheseseitig maximalen Optimierung flächenmäßig um 15,5 % gewachsen, die Komplexität der Design-Optimierung ist auch an dem Zuwachs der Anzahl an Netzen von 75 % zu erkennen. Durch Implementieren von DFT, Einfügen eines weiteren RAMs und durch die Geschwindigkeitsmaximierung des Designs konnten die auszufüllenden 20,6 % freie Die-Fläche (siehe Kapitel I) also optimal ausgenutzt werden.

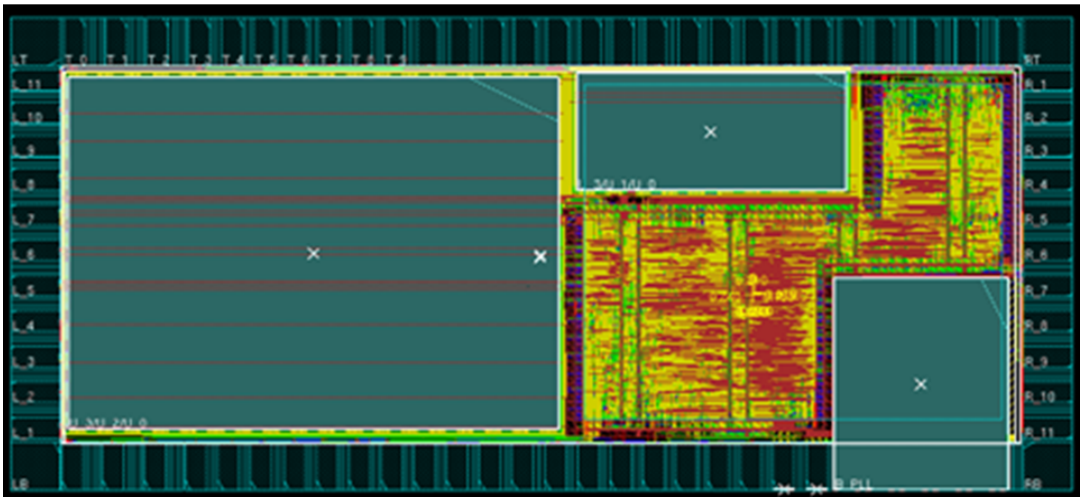


Abbildung 12: Ansicht vom Layout des ASIC-Designs „PDA-ASIC VERSION_2“ noch ohne Bond Pads in „Encounter“ von Cadence

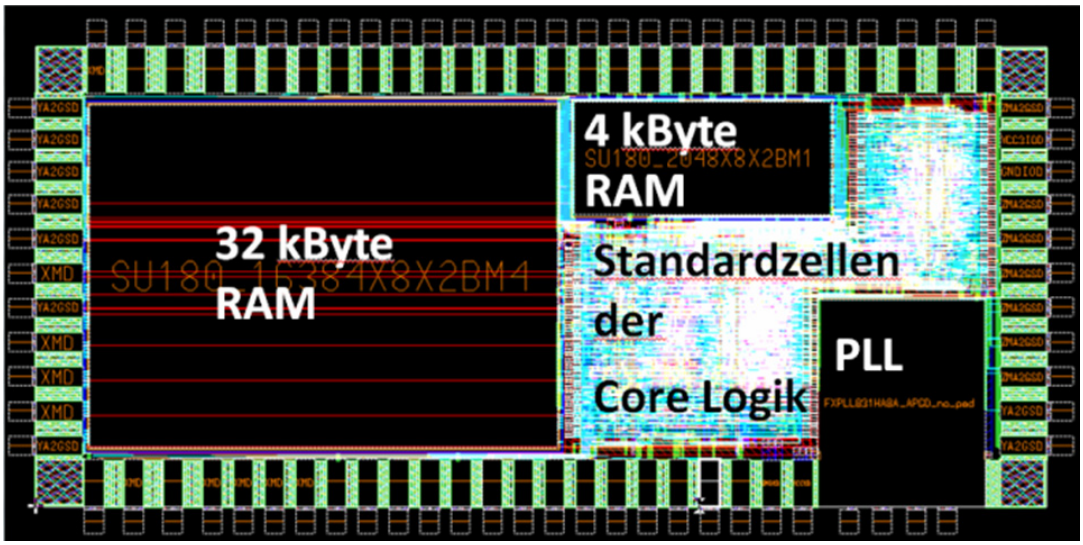


Abbildung 13: Ansicht des Layouts des ASIC-Designs „PDA-ASIC VERSION_2“ mit Bond Pads in der „IC Station“ von Mentor

Tabelle 5: Vergleich der Ergebnisse des Layouts „PDA-ASIC“ und des neuen „PDA-ASIC VERSION_2“

Design	Max. Taktrate gemäß Analyse von		Anzahl Logikzellen im Layout	
	Synthese	Layout	Exkl. Fillerzellen	Inkl. Fillerzellen
			Auffüllen des Floorplans →	
PDA-ASIC Version_2	177 MHz	243 MHz	20848 (vgl. Synthese: 16998 →Optimierung!)	28487
PDA-ASIC	72 MHz	100 MHz	8795 (vgl. Synthese: 9481 →Optimierung!)	24697

VI. Layout-Ergebnisse

Der Fokus soll hier neben der Synthese auch auf der Betrachtung des Layout-Ergebnisses liegen. Das Ergebnis nach dem Layouten in „Encounter“ von Cadence (siehe IC-Flow in Kapitel II) ist in Abbildung 12 dargestellt. Ein Import der GDS2-Datei des fertigen Layouts in die „IC-Station“ von Mentor folgt als nächster Schritt. Der Import führt mittels der automatisierten Implementierung der Bond Pads an den Pad Zellen zum finalen Layout (Abb. 13). Die Die-Fläche ist nun durch die Implementierung eines weiteren RAMs (4kByte) und durch die Geschwindigkeitsmaximierung des Designs im Vergleich zum ersten PDA-ASIC weitgehend genutzt. Die Bond Pads, welche beim Import eingefügt wurden, sind neben den Pad-Zellen zu erkennen.

Nach der bestmöglichen Optimierung der Synthese erzielt das neue PDA-Design eine maximal Taktrate von 177 MHz (Synthese) gegenüber den nur 72 MHz des alten Designs des „PDA-ASIC“ (Synthese).

Der Layout-Prozess ermöglicht eine weitere Optimierung des Designs (Tabelle 5), es werden zusätzliche Logikzellen im Layout eingefügt, um ein bestmögliches Timing-Resultat hinsichtlich der möglichen Taktrate erzielen zu können. Die maximal mögliche Taktrate steigert sich gemäß der Analyse des Layouts somit je nach Design („PDA-ASIC“ oder „PDA-ASIC VERSION_2“) nochmals um mehr als 37 %. Das neue PDA-Design kann nun nach „Encounter“ mit bis zu 243 MHz getaktet werden, dies stellt eine Erhöhung der maximalen Taktrate gegenüber dem ersten PDA-ASIC um 143 % dar (die zusätzliche Leistungsaufnahme durch die eingefügten Buffer ist noch zu prüfen).

Der reservierte Floorplan für die Platzierung der Standardzellen der Kernlogik ist im Vergleich zum Floorplan des ersten ASIC-Designs des „PDA-ASIC“ (Abbildung 3) signifikant vergrößert worden, um für das Erreichen der hohen Taktrate den nötigen Platz zur Verfügung stellen zu können. Der verbliebene Raum im Standardzellen-Floorplan wird mit Fillerzellen aufgefüllt. Deutlich ist festzustellen, dass der große Floorplan effizient ausgenutzt wird. Es werden bedeutend weniger Fillerzellen in das Layout eingefügt.

VII. Fazit

Design For Testability (DFT) konnte erfolgreich in das komplexe ASIC-Design des “PDA-ASIC VERSION_2” implementiert werden. Die Optimierungsziele der Leistungssteigerung konnten im neuen Design ebenfalls erreicht werden. Die verfügbare Fläche der Die konnte optimal genutzt werden durch Erweiterung des Designs um einen 4 kByte RAM auf nun 36 kByte RAM und durch eine Maximierung der Rechengeschwindigkeit des ASICs. Der neue PDA-ASIC wird nahezu 2,5-mal so schnell getaktet wie die Vorgängerversion und dabei, Dank der Ersparnis eines ganzen Taktes bei der Multiplikation, fast fünfmal so schnell multiplizieren können wie zuvor. Die Fertigung wird derzeit vorbereitet.

Danksagung

Besonderer Dank gilt Herrn Prof. Dr.-Ing. Dirk Jansen, welcher als Leiter des Instituts für Angewandte Forschung (IAF) der Hochschule Offenburg mit Forschungs- und Entwicklungsfreude mir möglich gemacht hat, diese Arbeit unter seiner Leitung durchzuführen. Weiterer Dank gilt den wissenschaftlichen Mitarbeitern Andreas Kreker und Sebastian Stickel, die beim Modifizieren des Quellkodes des Prozessorkerns und der Simulation wesentlich beteiligt waren.

Literaturverzeichnis

[1] Dipl.-Ing. Marc Durrenberger, Dipl.-Ing. Daniel Bau, Prof. Dr.-Ing. D. Jansen, „SIRIUS Mikrocontrollersystem Design Kit“, *Technischer Bericht*, HS-Offenburg, 2007

[2] D. Bau, „Development of an Electro Physiological Front End with SPI-Bus Connection”, *Hochschule Ulm: MPC-Workshopband 40*, 2008

[3] Faraday, „FSA0A_C_Generic_Core_Specification_v1.1, FSA0A_C 0.18µm Generic Core Cells” , 2004

[4] D. Bau, „Der UMC 0.18 Design Flow am Beispiel eines PDA-Prozessor ICs“, *Hochschule Ulm: MPC-Workshopband 42,* 2009

[5] Synopsys, „Datasheet DW02_mult_2_stage Two-Stage Pipelined Multiplier”, 2011

[6] Benjamin Dusch, „Entwicklung einer Boundary Scan Architektur für ASIC-Design, Band 1- Dokumentation“, *Bachelor –Thesis*, HS-Offenburg, 2010

[7] B. Dusch, „Design For Testability (DFT) Strukturen für ASIC-Design und ihre Emulation auf FPGA“, *Hochschule Ulm: MPC-Workshopband 44,* 2010

[8] IEEE, „Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1™-2001 (R2008)”, New York, 2008

Benjamin Dusch erhielt den akademischen Grad des B. Eng. in Elektro- und Informationstechnik im Jahr 2010 von der Hochschule Offenburg und studiert derzeit im Masterprogramm. Er ist wissenschaftlicher Mitarbeiter an der Hochschule Offenburg.

Design and Test of a Gigabit Ethernet MAC for High-Speed HIL-Support

Alexander Rohleder, Steffen Jaeckel, Axel Sikora

***Abstract*—The efficient support of Hardwae-In-the-Loop (HIL) in the design process of hardware-software-co-designed systems is an ongoing challenge. This paper presents a network-based integration of hardware elements into the software-based image processing tool „ADTF“, based on a high-performance Gigabit Ethernet MAC and a highly-efficient TCP/IP-stack. The MAC has been designed in VHDL. It was verified in a SystemC-simulation environment and tested on several Altera FPGAs.**

***Index Terms*—Hardware-Software Co-Simulation, Hardware In the Loop (HIL), Gigabit Ethernet MAC, VHDL, SystemC.**

I. INTRODUCTION

The design of complex hardware-software-co-design systems mostly starts with its behavioural and high-level description in software [1] [2]. The subsequent process of stepwise implementation and verification is an ongoing challenge, as high-level software blocks must be executed in combination with Hardware-In-the-Loop (HIL) elements. Existing solutions often are proprietary, requiring dedicated hardware and software interface, as e.g. [3].

The development of advanced driver assistance systems (ADAS) is a key element for modern and safe vehicles. These systems may use contact- or non-cooperative perception sensor systems for the detection, such as CMOS-cameras, RADAR or LIDAR. The subsequent processing steps might have a very high complexity. This will be especially the case, if the objects, which shall be detected, have a variety of possible classifiers. And this is the case for predictive pedestrian protection systems (PPPS) [4], as they are recently available from some car manufacturers.

However, these systems will find a broad dissemination, if their cost in mass production is low enough, which calls for hardware-assisted solutions. One example was presented in [5].

Alexander Rohleder, alexander.rohleder@stzedn.de, Steffen Jaeckel, steffen.jaeckel@stzedn.de, Axel Sikora, axel.sikora@stzedn.de, Poststr. 35, 79423 Heitersheim, Germany

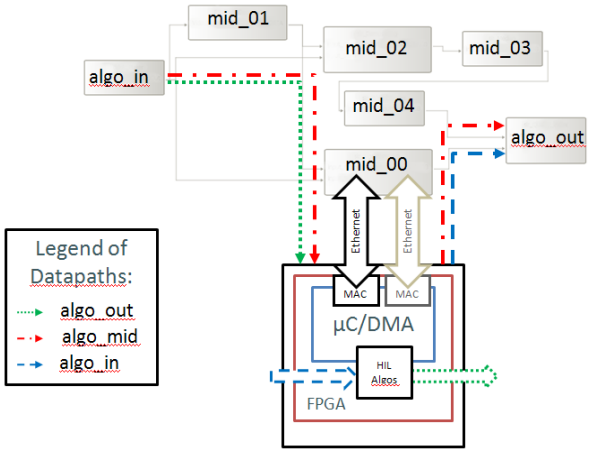


Fig. 1 Integration of hardware modules into ADTF

II. THE HIL APPROACH

A. The Basic Framework

For the development of advanced driver assistance systems (ADAS), common frameworks are used by practically all European OEMs and 1st tier suppliers, which allow an easy combination of processing blocks being described in high-level software, and is a valuable tool for the evaluation of different algorithms and their suitability for real input streams, i.e. video streams from the CMOS cameras.

After the algorithms are evaluated and approved, system developers typically start to migrate the functional blocks successively into firmware or hardware implementations.

The verification of the system is only possible after the process of full migration to the productive system. There exists no standardized possibility for early integration of hardware elements into the framework.

Especially the “Automotive Data and Time triggered Framework” (ADTF) is widely used. The restrictions described above are not limited to the ADTF framework, but are a limitation of many simulation frameworks. Therefore, the proposed approach can be regarded as a general concept.

B. Extension to the Framework

Within the ADTF framework, it is possible to link several ADTF instances via the so called Messagebus protocol with the objective to distribute highly complex implementations on several workstations in order to improve real-time characteristics. The Messagebus protocol is a generic application layer protocol, accessing the network resources over a standard TCP/IP-socket interface. It can potentially be used for the integration of dedicated hardware blocks into the flow.

C. Requirements

The integration of dedicated hardware blocks is possible in three different positions, as shown in Fig. 1:

- algo_in: In this case the dedicated hardware block is positioned as first block in the data path and can provide the link to a camera or a player system,
- algo_out: In this case the dedicated hardware block is positioned as last block in the data path fand can provide the interface to a display or analysis system.
- algo_mid: In this case, the dedicated hardware block is positioned in the middle of the data path. As this case encompasses the two other cases, and represents the typical use case, it is the one to concentrate on.

For the maximum bandwidth requirement, a full-colour (RGB) image sensor with a sensitivity of 12bit, a resolution of 1024 x 512 and a frame rate of 43 frames / s is assumed as algo_in block. This calculates to a net data rate of around 820 Mbit / s, where the exact data rate depends on the control information, which is included into the data stream.
It is anticipated that all subsequent blocks work on images with less information, e.g. edge images, regions of interest, or alike, so that this number is regarded as maximum. A Gigabit Ethernet interface is therefore regarded as most suitable network connectivity, as it fulfils the bandwidth requirements and allows an easy use on the PC side. However, some additional precautions have to be taken to sustain the bandwidth on the hardware system.

III. The MAC Architecture

A. General Architecture

A Gigabit Ethernet MAC has been designed, which follow the legacy partitioning into three main parts, the MAC core, the PHY interface and the bus interface. These three parts can be integrated into existing systems as one instance of the MAC.

As shown in Fig.2 both the PHY and the bus interface are connected via FIFO interfaces, allowing easy replacement by implementations for different PHY, and bus interfaces, respectively.

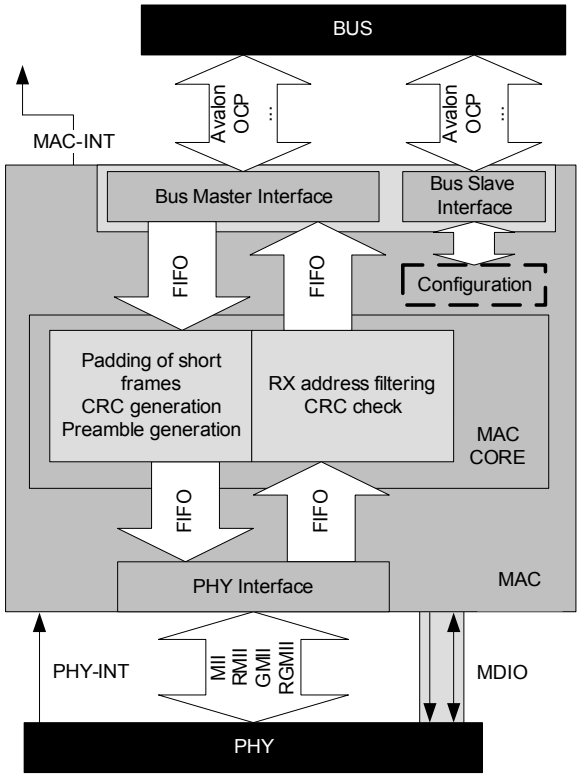


Fig.2 Block diagram of the Gigabit Ethernet MAC

Every single task, e.g. CRC checksum verification, address filtering, etc. is implemented as a single block, which brings the advantage that:

- Every single module is optional. E.g. if the VLAN tag support is not needed, it can just be omitted.
- Easy extensibility allows the integration of special function blocks. An example is the checksum calculation for higher-layer protocols.

The detailed block diagrams of the transmit and the receive paths are shown in Fig. 3 and Fig. 4, respectively.

B. Bus Interfaces

As the designed block shall be used flexibly in different environments, also the bus interfaces shall be interchangeable. I.e. the following bus interfaces shall be available in the first version:

- Avalon is the bus system used on Altera Nios II SoCs, called SOPC (System on programmable Chip). This interface is currently implemented conforming to version 1.3.
- AXI is one of five buses/interfaces defined in the Advanced Microcontroller Bus Architecture (AMBA) by ARM ltd. where AXI stands for Advanced eXtensible Interface. The version that shall be supported is AXI 2.0

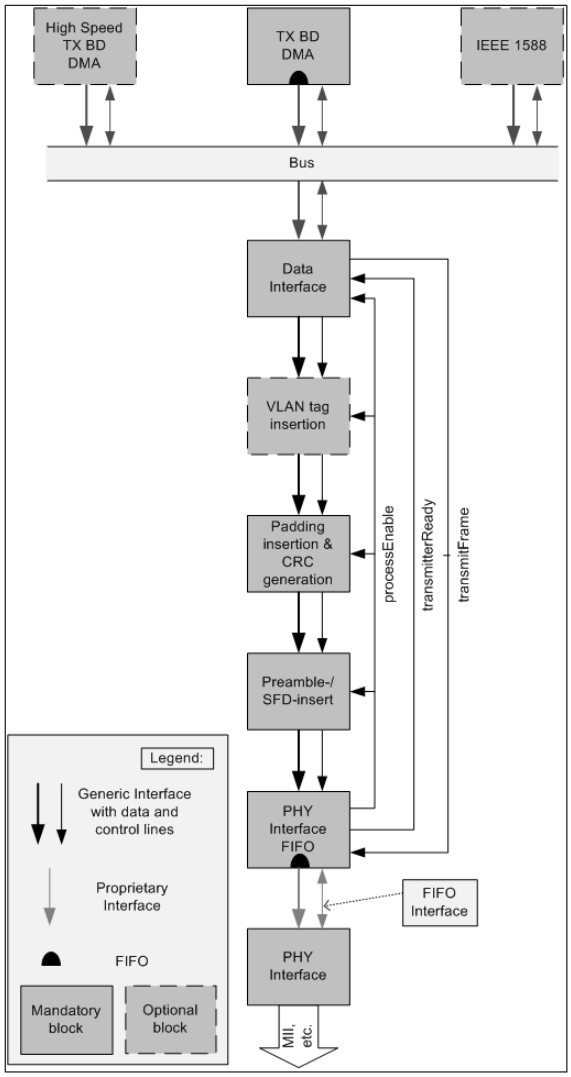


Fig. 3: Diagram of modular block architecture for the transmit data path

- OCP (Open Core Protocol) is a family of bus interfaces that is defined by the OCP International Partnership. It is not proprietary, openly licensable and an extensive and very comprehensive specification. The design will implement a sub-set of OCP version 3.0.

It is planned that all the three bus interfaces will be implemented as Single-Request-Multiple-Data Master and Single-Request-Single-Data Slaves, where both are supporting only In-Order requests and serving In-Order responses.

IV. Design, Test and Implementation

A. Design Principles

The complete design shall be usable both in hard-wired ASICS and programmable logic devices (PLD).

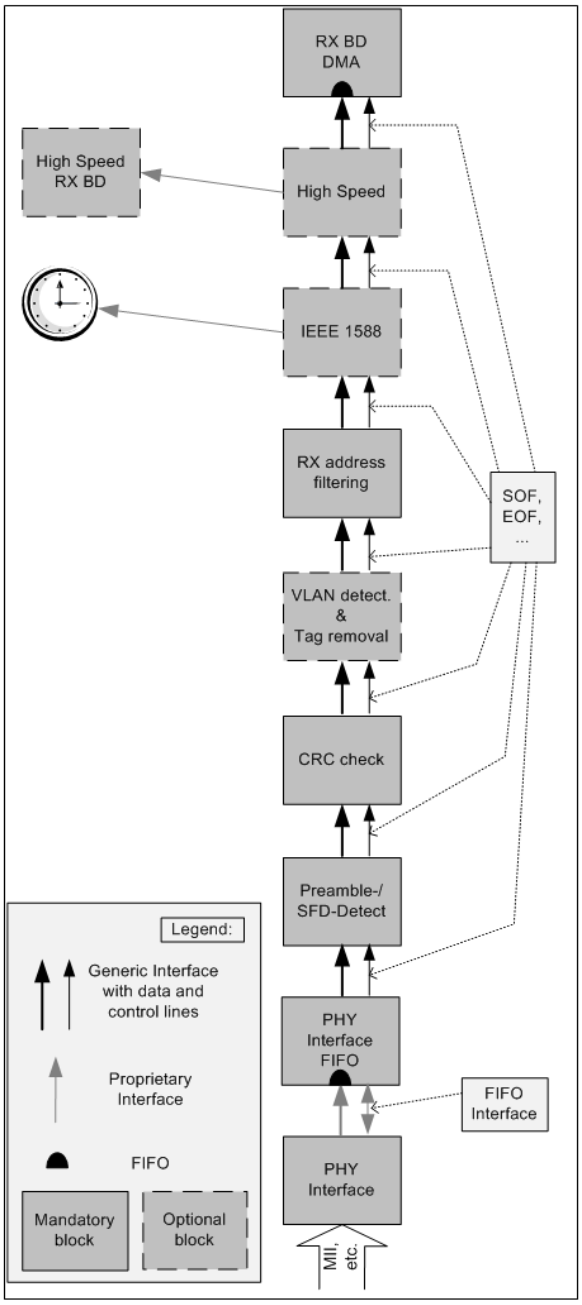


Fig. 4: Diagram of modular block architecture for the receive data path

It is therefore described in finite state machines using VHDL. In order to achieve good portability, the memory blocks are abstracted with a generic “wrapper”.

B. Test Environment

The verification of hardware components is always one of the most time-consuming tasks in the development process. As stated in [6], the full verification of the behaviour of a HDL model may consume as much as 60 to 80 % of the efforts.

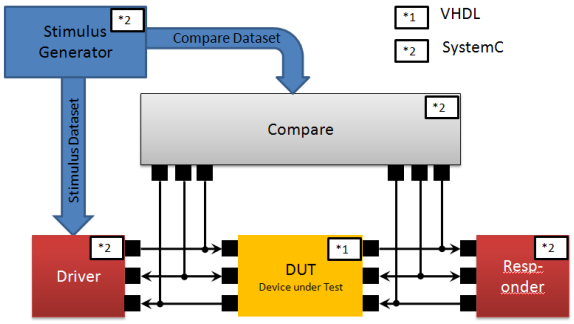


Fig. 5: Simulation Environment I - Module Tests

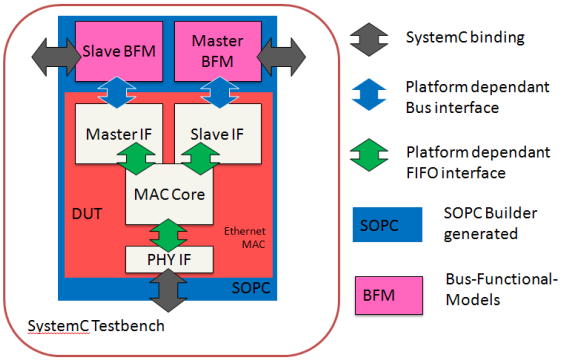


Fig. 6: Simulation Environment II - Full-System Simulation

There exist already several approaches like the "random constraints" included for example in the SystemVerilog language since version 3.1 to easily accomplish the task of full verification [7]. The missing experience with SystemVerilog and Verilog in the design team led to an implementation of a similar framework in SystemC for testing HDL models implementing packet based communication devices. This framework follows the UNH IOL[1] test cases, a testing and functional verification facility for Ethernet networking devices.

Mentor Graphics Questa was used as simulation engine, which allows cross-language simulation between VHDL, Verilog, SystemVerilog and SystemC models.

The simulation environment is shown in Fig. 4 for the module tests, and in Fig. 5 for the full system.

C. Implementation & Verification

The implementation followed a requirements engineering design approach and started with a detailed description of some 65 requirements.

In the second step the complete design and simulation environment was prepared, including the combined use of VHDL in the design itself, and SystemC for the simulation and verification environment.

The subsequent design phase started with the partition, and included the implementation and verification. As a result, around 10k lines of code (loc) for the VHDL design were generated. The verification environment includes around 2k loc of the SystemC framework (C++), and 12k loc for the simulation and verification (10k C++, 2k SystemVerilog).

Apart from the behavioural verification during the design phase, the network block was extensively tested in real networking environments.

D. Results

The design was first implemented in various Altera Cyclone II and III devices. In both devices, the MAC with simple FIFO Interface consumes ~1800 LUTs and ~1400 registers. The MAC with Avalon bus interface and statistical functions requires ~7100 LUTs and ~5100 registers. For verification, a NIOS host processor was used together with an embedded TCP/IP stack from the authors' institute.

V. Summary and Outlook

This project opened the opportunity to improve existing work-flows and to integrate new approaches. The implementation of a requirements engineering-based approach and a distributed development process was a challenge for every contributing engineer and will improve the flow of the future projects at the authors' institute. The design itself could be fruitfully used already also in further projects.

Acknowledgement

The presented work results from the joint project "Propedes" (Predictive Pedestrian Protection at Night), which is part of the project initiative eNOVA and is partially funded by the Federal Ministry of Education and Research (BMBF) under contract number 13N9754.

References

[1] D. Jansen (Ed.), "Handbuch der Electronic Design Automation", Fachbuchverlag Leipzig, 2001.

[2] A. Sikora, R. Drechsler, "Software-Engineering und Hardware-Design: Eine systematische Einführung", *Carl Hanser Verlag,* 2002.

[3] O. Gietelink, J. Ploeg, B. De Schutter, M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations", *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, Vol. 44, Issue 7, 2006.

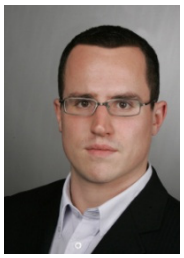
[4] D. Gerónimo, A.M. López, A.D. Sappa, T. Graf, "Survey of Pedestrian Detection for Advanced Driver Assistance Systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 2010.

[5] S. Jaeckel, A. Sikora, W. Rülling, "Implementierung eines Single Pass Connected Component Labeling Algorithmus zur Detektion von leuchtenden Objekten in Nachtszenen im Automotive Umfeld", *43. Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg*, Göppingen, 2010.

[1] UNH IOL = University of New Hampshire InterOperability Laboratory
http://www.iol.unh.edu/services/testing/

[6] J. Bergeron, “Writing Testbenches: Functional verification of HDL models”, Kluwer Academic Publishers, 2000.

[7] Accellera Organization Inc., “SystemVerilog 3.1 - Extensions to IEEE 1364-2001 Verilog”, 2003.



Steffen Jaeckel studied Computer Engineering at the University of Applied Sciences, Furtwangen and received his Dipl.-Inform. degree in April 2009. After that, he continued to work as project engineer at Steinbeis Innovation Center Embedded Design and Networking mainly in the field of embedded wireless and wired communication, embedded security and hardware design for networking protocols.

Alexander Rohleder studied Information Technology at the Baden-Wuerttemberg Cooperative State University Loerrach and received his B.Eng. degree in Sep. 2010. His bachelor thesis was on the protocol implementation of Messagebus protocol in SW and HW. Since then, he works as project engineer at Steinbeis Innovation Center Embedded Design and Networking mainly in the field of embedded wireless communication.

Axel Sikora was awarded the Dipl.-Ing. in electrical engineering in 1993 and the Dipl. Wirtsch.-Ing. in 1995 from RWTH Aachen. He earned his Ph.D. at Fraunhofer Institute IMS at Gerhard-Mercator-University Duisburg. Since 1999, he is Professor at the Baden-Wuert¬temberg Cooperative State University Loerrach, where he leads the Steinbeis Innovation Center Embedded Design and Networking.

Entwurf eines dynamisch rekonfigurierbaren Rechensystems auf Basis eines Virtex-5 FPGAs

André Netzeband, Frank Kesel, Thomas Greiner

***Zusammenfassung*— Dynamisch rekonfigurierbare Rechensysteme sind Systeme, welche ihre Hardwarekonfiguration zur Laufzeit verändern können. Hiermit ist es möglich sehr flexible Rechensysteme aufzubauen, welche im Bezug auf Energieverbrauch, Chipfläche und Kosten dennoch effizient sind. Im Gegensatz zu Standardrechensystemen, bei denen eine hohe Flexibilität durch überdimensionierte Rechenleistung erreicht wird, setzen dynamisch rekonfigurierbare Rechensysteme auf Hardwareeinheiten, die komplexe Berechnungen parallel ausführen und bei einer neuen Aufgabenstellung ausgetauscht werden können. Die Realisierung solcher Systeme ist mit partiell rekonfigurierbaren FPGAs möglich, die es zulassen nur einen Teil ihrer Logik zu rekonfigurieren, während die restliche Logik in ihrer Funktion nicht unterbrochen wird.**

Dieser Beitrag beleuchtet die Grundlagen solcher Rechensysteme, zeigt eine mögliche Realisierung mit einem Virtex-5 FPGA auf und beschreibt eine Softwareabstraktion, welche es einem Linux-Betriebssystem erlaubt, die Hardware im System automatisch an die Erfordernisse der Anwendungsprogramme anzupassen. Am Ende wird ein Hardware-Scheduler vorgestellt, mit dem es möglich ist komplexe Aufgabenstellungen, die das Zusammenspiel mehrerer Hardwareeinheiten erfordern, zeitlich gestaffelt abzuarbeiten. Hierbei steht das Ziel im Vordergrund die Rekonfiguration ausschließlich vom Betriebssystem durchführen zu lassen, so dass deren Komplexität vor einem Anwendungsprogramm verborgen bleibt.

***Schlüsselwörter*—FPGA, Rekonfigurierung, dynamisch, partiell, run-time reconfiguration, reconfigurable computing, Linux, Hardware Scheduler.**

I. EINLEITUNG

Der Entwurf eines Rechensystems kann grundsätzlich auf zwei Arten stattfinden: Einerseits kann ein universelles Rechensystem entworfen werden, das aufgrund einer Überdimensionierung (hohe Taktraten, viel Speicher, viele Hardwarebeschleuniger etc.) in der Lage ist, flexibel mit den unterschiedlichsten Aufgabenstellungen umzugehen. Andererseits lässt sich ein Rechensystem auch aufgabenbezogen entwerfen, so dass es eine konkrete Aufgabe besonders effizient, zum Beispiel durch parallele Berechnungen, durchführen kann. Solange die einzige physikalische Realisierungsform einer umfangreichen digitalen Schaltung die Fertigung in Form eines ASICs[1] war, konnten solche spezialisierten Rechensysteme nur dann preiswert hergestellt werden, wenn der dabei entwickelte ASIC später in großen Stückzahlen verbaut werden konnte. Je spezieller jedoch eine Aufgabenstellung ist, desto seltener muss diese auch erfüllt werden, was wiederum dazu führte, dass es sich kaum lohnte, spezialisierte Rechensysteme als ASIC anfertigen zu lassen. Daher ist für lange Zeit der übliche Ansatz zur Lösung solcher Probleme ein universelles Rechensystem gewesen, welches leistungsstark genug war, um ein breites Spektrum an Aufgabenstellungen zu erfüllen. Im Bereich der Signalverarbeitung setzten sich somit sogenannte digitale Signalprozessoren (DSPs) durch.

Mit der Einführung von Field Programmable Gate Arrays (FPGAs) in den 80er Jahren änderte sich diese Situation jedoch grundlegend. Nun war es möglich, digitale Schaltungen, die nur in geringen Stückzahlen realisiert werden sollten, in solchen FPGAs zu implementieren. Da es sich bei FPGAs wiederum um Standardbausteine handelt, lag der Preis für einen FPGA in einer ähnlichen Größenordnung wie der eines Standardprozessors. Durch die Möglichkeit nahezu beliebige digitale Schaltungen in solchen FPGAs implementieren zu können, konnten ganze Berechnungsalgorithmen für spezielle Aufgabenstellungen in solchen Bausteinen untergebracht werden, was zu einer weiten Verbreitung im Bereich der Signalverarbeitung führte. Mit Hilfe von FPGAs konnten also spezialisierte Rechensysteme erstmals kostengünstig umgesetzt werden. Das große Problem dabei war allerdings die mangelnde Flexibilität des Systems: Änderte sich die Aufgabenstellung auch nur geringfügig, so konnte diese mit der spezialisierten Schaltung nur schwer oder eventuell auch gar nicht mehr bewältigt werden. Da

André Netzeband, andre@netzeband.eu,
Prof. Dr.-Ing. Frank Kesel, frank.kesel@hs-pforzheim.de,
Prof. Dr.-Ing. Thomas Greiner, thomas.greiner@hs-pforzheim.de,
Hochschule Pforzheim, Fakultät Technik, Tiefenbronner Str. 65,
75175 Pforzheim

[1] ASIC = Anwendungsspezifische Integrierte Schaltung.

ein FPGA normalerweise nur beim Start einer Baugruppe konfiguriert wird und dann während der gesamten Systemlaufzeit diese Konfiguration beibehält, konnten sich solche Systeme auf zur Laufzeit wechselnde Aufgabenstellungen nicht flexibel genug anpassen. Wurde dies jedoch von einem System verlangt, blieb weiterhin nur die Lösung eines überdimensionierten und damit universellen Rechensystems.

Seit der Virtex-II Generation bietet der FPGA-Hersteller Xilinx Bausteine an, die sich zur Laufzeit *partiell rekonfigurieren* lassen. Darunter ist die Änderung der Logikkonfiguration eines Teils des FPGAs zu verstehen, währenddessen der restliche Teil des FPGAs vollständig funktionsfähig bleibt und in seiner Arbeit nicht unterbrochen werden muss. Mit solchen FPGAs ist es möglich, Rechensysteme zu entwerfen, welche ihre Hardwarekonfiguration zur Laufzeit an die jeweilige Aufgabenstellung anpassen können: Sogenannte *dynamisch rekonfigurierbare Rechensysteme*. Die Veränderung der Hardwarekonfiguration eines solchen Systems im zeitlichen Verlauf kann dabei als eine Art *Hardware-Zeitmultiplex* betrachtet werden, was dann wiederum zu einer *virtuellen Vergrößerung* der verfügbaren *Chipfläche* führt (vgl. [1]). Solche Systeme besitzen einerseits die hohe Flexibilität, die sonst nur universelle Rechensysteme (General Purpose Processors – GPPs) bieten, und andererseits lassen sie sich in Bezug auf Chipfläche, Energieverbrauch und Kosten fast genauso effizient umsetzen wie ein auf eine konkrete Aufgabenstellung spezialisiertes Rechensystem.

II. Technische Realisierung

A. Umsetzungsvarianten

Das Thema „*rekonfigurierbare Rechensysteme*“ (*reconfigurable computing*) ist schon in mehreren Publikationen und im Rahmen diverser Forschungsarbeiten [1]-[7] untersucht worden. Hierbei zeigt sich, dass die dynamische Rekonfiguration von Rechensystemen auf unterschiedlichen Abstraktionsstufen stattfinden kann. In [3] wird in diesem Zusammenhang zwischen fein- und grobgranularer Rekonfiguration gesprochen, wobei sich diese Begrifflichkeiten vor allem auf das Umschalten von Datenpfaden und deren Breite zwischen matrixartig angeordneten Verarbeitungseinheiten bezieht. Damit umfasst diese Einteilung nicht das gesamte Spektrum möglicher Realisierungsformen, weswegen an dieser Stelle eine Unterteilung nach dem Beispiel des in [8] und [9] vorgestellten Y-Diagramms (siehe Abbildung 1) erfolgen soll. Hierbei lassen sich gängige Umsetzungen dynamisch rekonfigurierbarer Rechensysteme auf *algorithmischer Ebene*, *Register-Transfer-Ebene* und *Gatterebene* finden.

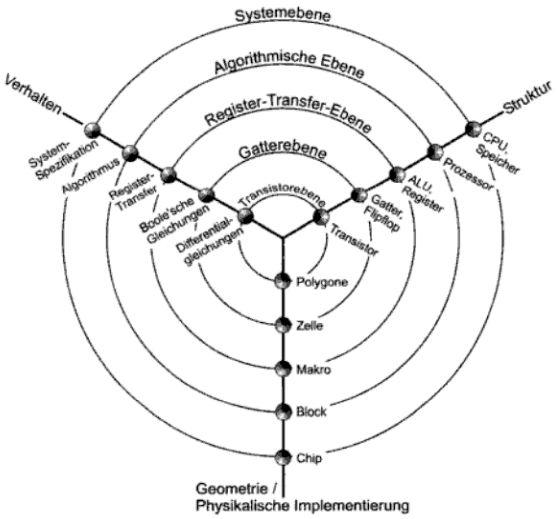


Abbildung 1: Das Y-Diagramm nach Gajski [8] beschreibt die Sichtweisen beim Entwurf digitaler Schaltkreise (Verhaltensbeschreibung, Strukturbeschreibung, geometrische Beschreibung) im Kontext des jeweiligen Abstraktionsgrades (von der Systemebene bis zur Transistorebene).

1) Algorithmische Ebene

Bei Rechensystemen, die sich auf algorithmischer Ebene dynamisch rekonfigurieren lassen, werden komplette Subsysteme, wie beispielsweise Coprozessoren, rekonfiguriert. Die grundlegende Struktur des Systems und die Datenpfade zwischen den Subsystemen bleiben hierbei erhalten. Der Vorteil einer solch groben Rekonfiguration ist, dass sich herkömmliche Rechensysteme leicht um eine solche Funktion erweitern lassen und weiterhin die gleichen Entwicklungswerkzeuge und Compiler für die Programmierung dieser Systeme benutzt werden können. Allerdings sind solche Systeme weniger flexibel, da sowohl die Anzahl und Art möglicher dynamisch rekonfigurierbarer Hardwareeinheiten als auch deren Position und Größe im FPGA zur Entwurfszeit des Systems festgelegt werden müssen. Es lassen sich zwar später neue Hardwareeinheiten hinzufügen, doch für diese ist jedes Mal ein erneuter Teilentwurf nötig. Beispiele für solche Systeme sind die Erlangen Slot Machine (vgl. [2], S. 51ff.), das VAPRES-Projekt [7] oder das DPR-Projekt der Hochschule Pforzheim [1].

2) Register-Transfer-Ebene

Sind Rechensysteme auf Register-Transfer-Ebene rekonfigurierbar, so werden vor allem Datenpfade zwischen Registern und Schaltnetzen angepasst oder ganze Schaltnetze gegeneinander ausgetauscht. Eine solche Rekonfiguration wird in der Literatur häufig im Zusammenhang mit Prozessoren erwähnt, welche aus matrixartig angeordneten Verarbeitungseinheiten (z.B. ALUs) bestehen. In [3] werden solche Systeme als *grobgranular* bezeichnet („*coarse grained*“). Hierbei lassen sich durch die Veränderung von Datenpfaden

einer Aufgabe mehrere parallele und sequenzielle Rechenwerke zuteilen und unterschiedlichen, parallel abzuarbeitenden Aufgaben räumlich getrennte Bereiche im „Matrix-Prozessor“ zuordnen. Ein Beispiel ist hier das PipeRench-Projekt [4].

3) Gatterebene

Rekonfiguration auf Gatterebene betrifft die direkte Anordnung und Verbindung von logischen Gattern. Eine klare Abgrenzung zu den anderen Arten der Rekonfiguration kann hier jedoch nur schlecht gelingen, da die Übergänge von der Register-Transfer-Rekonfiguration zur Gatterrekonfiguration entweder fließend sind oder es sich um Mischformen verschiedener Varianten handelt. So stellt sich die Frage, ob eine *feingranulare* („*fine grained*“) Rekonfiguration – bei der anstatt kompletter Signalvektoren (z.B. 32 Bit) nur noch einzelne Signale (1 Bit) umgeschaltet werden – schon einer Rekonfiguration auf Gatterebene entspricht. Bei solchen Projekten werden ebenfalls häufig matrixartige Anordnungen von Verarbeitungseinheiten verwendet. Allerdings sind diese Einheiten nur noch wenige Gatter groß und verarbeiten jeweils nur noch ein Bit. Ein Projekt in diesem Bereich ist die Garp-Architektur (vgl. [3], S. 30ff.). Außerdem tritt eine Rekonfiguration auf Gatterebene in diversen Projekten auch im Zusammenhang mit einer Rekonfiguration auf algorithmischer Ebene auf. Hierbei werden die Bitdatenströme, welche die dynamisch rekonfigurierbaren Hardwareeinheiten auf algorithmischer Ebene beschreiben, so manipuliert, dass sich einzelne „Gattereigenschaften“ verändern lassen. Somit ist es möglich, eine Hardwareeinheit geringfügig zu verändern, um beispielsweise die Koeffizienten eines Filters anzupassen oder die Position der Hardwareeinheit in einem FPGA zu beeinflussen. Beispiele für solche Bitmanipulationen sind in [5] und [6] zu finden.

Dieser Beitrag behandelt im weiteren Verlauf nur noch die dynamische Rekonfiguration auf algorithmischer Ebene.

III. Realisierung im FPGA

Wie schon im einleitenden Kapitel angesprochen, können dynamisch rekonfigurierbare Rechensysteme mit partiell rekonfigurierbaren FPGAs realisiert werden. Hierbei werden, wie in Abbildung 2 zu sehen, mit dem Programm PlanAhead beim sogenannten Floorplan-Prozess eine beliebige Anzahl von partiell rekonfigurierbaren Regionen (im Xilinx-Sprachgebrauch: Partitionen) in ihrer Position und Größe auf der FPGA-Fläche definiert und ihnen die Netzlisten der dazugehörigen Hardwaremodule zugeordnet.

Die einer Partition zugeordneten Hardwaremodule müssen jeweils die gleiche Schnittstelle zum statischen Teil der Hardware besitzen. Im Anschluss werden neben den Bitdaten für das Gesamtsystem auch

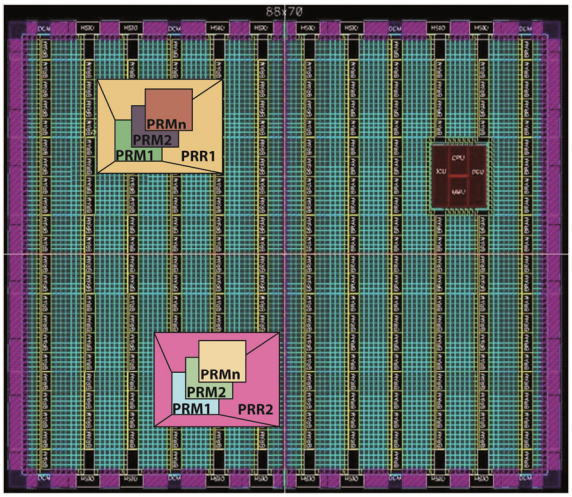


Abbildung 2: Die schematische Darstellung eines Virtex-4 FPGAs mit zwei partiell rekonfigurierbaren Regionen (PRR1 und PRR2). Diese Regionen lassen sich mit einem partiellen Bitstream rekonfigurieren, ohne dass die Logik um sie herum (die statische Logik) in ihrer Funktion unterbrochen wird (Bildquelle: Xilinx).

die partiellen Bitdaten zum Rekonfigurieren der einzelnen Partitionen erzeugt. Diese partiellen Bitdaten lassen sich während der Laufzeit des Systems über die externe oder interne Konfigurationsschnittstelle in das FPGA laden.

IV. Zielapplikationen

Die Vorteile dynamisch rekonfigurierbarer Rechensysteme werden vor allem in den Anwendungen deutlich, die einerseits eine hohe Flexibilität und andererseits eine hohe Effizienz im Bezug auf Energieverbrauch, Chipfläche und Kosten erforderlich machen. Seit Mitte 2009 befinden sich sogenannte Multimedia-Center auf dem Markt, welche nahezu beliebige Medienformate (Video- und Audiodaten) von einer Festplatte oder direkt aus dem Internet abspielen können. Die hier benötigte Flexibilität wird in solchen Geräten überlicherweise mit Hilfe eines GPP (General Purpose Processor) sowie einem DSP (Digital Signal Prozessor) erreicht. Anstatt dieser Zweikernarchitektur ließen sich solche Systeme auch mit dynamisch rekonfigurierbaren Hardwaredecodern realisieren, was sich positiv auf die Chipfläche auswirken könnte.

Bei der Übertragung digitaler Videodaten über Satellit, Kabel, terrestrischen Funk oder das Internet gibt es hingegen fest definierte Verfahren zur Decodierung und eventuellen Entschlüsselung der Daten. Hierbei sind diese Verfahren jedoch unabhängig vom eigentlichen Inhalt (MPEG-2 für SDTV und MPEG-4 für HDTV) und damit immer ein Kompromiss zwischen Bandbreitenverbrauch und Videoqualität. Der Artikel [10] beschreibt, wie sich eine auf das jeweilige Bild- und Audiomaterial angepasste En- und Dekodierung realisieren ließe und verweist darauf, dass hiermit die zur Verfügung stehende Bandbreite bei zugleich höherer Ton- und Bildqualität besser ausgenutzt werden

könnte. Auch in diesem Zusammenhang würden dynamisch rekonfigurierbare Rechensysteme die hierzu nötige Flexibilität bieten, ohne dass auf überdimensionierte DSPs oder GPPs zurückgegriffen werden müsste.

Ein weiteres mögliches Einsatzfeld solcher Rechensysteme wäre das Software Defined Radio. Hierbei handelt es sich um ein Forschungsprojekt, das militärische Funkgeräte in die Lage versetzen soll, eine Vielzahl von Dekodier-, Verschlüsselungs- und Übertragungsverfahren in einem breitbandigen Frequenzspektrum zu unterstützen. Es soll damit verbündeten Streitkräften ermöglicht werden miteinander zu kommunizieren und den Übertragungskanal, entsprechend der Begebenheiten am jeweiligen Einsatzort, ideal auszuwählen. Um diese Geräte in einem möglichst breitbandigen Frequenzbereich anpassen zu können, ist eine Signalverarbeitung mit sehr hohen Datenraten nötig. FPGAs sind aufgrund der Möglichkeit zur Parallelisierung von Berechnungen modernen DSPs in dieser Hinsicht häufig überlegen, weswegen sich eine FPGA-basierte Signalverarbeitung gerade beim Software Defined Radio anbieten würde. Mit Hilfe dynamsich rekonfigurierbarer Signalverarbeitungsalgorithmen wäre ein solches System, trotz höherer Effizienz, genauso flexibel einsetzbar wie ein DSP-basiertes System.

V. Implementierung

A. Hardwarekonfiguration

In diesem Kapitel wird eine mögliche Implementierung eines dynamisch rekonfigurierbaren Rechensystems aufgezeigt. Hierbei basiert das zugrundeliegende System auf dem DPR-Projekt (dynamisch partielle Rekonfiguration), das im Rahmen einer Masterthesis an der Hochschule Pforzheim angefertigt wurde. Dieses Projekt ist im Artikel [1] ausführlich beschrieben. Ein Ergebnis dieser Arbeit war die Einführung eines Hardwarebeschleunigers, mit dem die Rekonfiguration unabhängig vom Prozessor durchgeführt wurde und welcher gleichzeitig die Dauer der Rekonfiguration ungefähr um den Faktor 1000 gegenüber dem reinen Softwareansatz von Xilinx verringerte (vgl. [1]).

Dieser Hardwarebeschleuniger, der im Folgenden als NPI-Modul bezeichnet wird, ist auch in dem hier beschriebenen Projekt verwendet worden. Diesem Modul wird die Startadresse und Größe der partiellen Bitdaten mitgeteilt, woraufhin es diese Daten automatisch aus dem Speicher lädt und die Rekonfiguration vornimmt. Die Dauer der Rekonfiguration hängt hierbei von der Größe der partiellen Bitdaten ab, die wiederum von der Größe der definierten dynamisch rekonfigurierbaren Partitionen abhängt. Bei der im vorliegenden Projekt benutzten Partitionsgröße (11 × 20 CLBs[2]) beträgt die Rekonfigurationsdauer in etwa 170 µs.

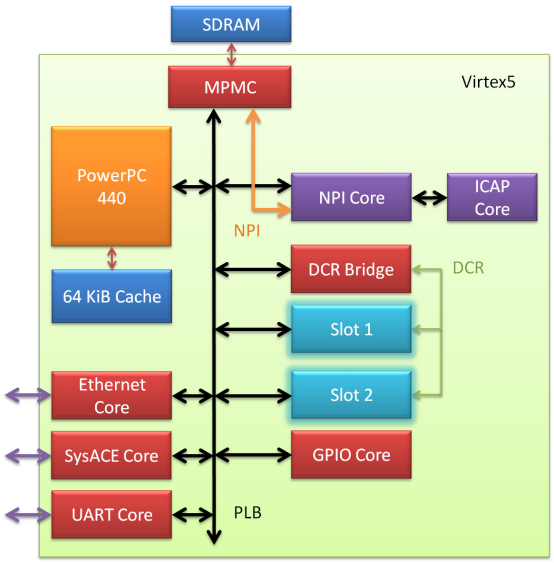


Abbildung 3: Das im vorliegenden Projekt verwendete Rechensystem: Neben einer PowerPC 440 CPU verfügt das System über mehrere Ein- und Ausgabeeinheiten sowie den NPI-Core und ICAP-Core, welche für die Rekonfiguration zuständig sind. Die dynamisch rekonfigurierbaren Bereiche sind innerhalb der Slots enthalten, von denen zwei vorhanden sind. Diese Slots werden über einen DCR-Bus gesteuert, auf den mit Hilfe einer DCR-Bridge vom Prozessor aus zugegriffen werden kann.

Abbildung 3 zeigt den grundlegenden Aufbau des im vorliegenden Projekt verwendeten Rechensystems. Als Prozessor wird eine Power PC 440 CPU eingesetzt, welche im Virtex-5 FPGA (XC5VFX70T) als eingebetteter Prozessor enthalten ist. Das System verfügt neben zahlreichen Ein- und Ausgabeeinheiten (Ethernet, UART, CF-Card-Interface) über 256 MiB externem DDR2-SDRAM und den zur Rekonfiguration nötigen NPI-Core. Der NPI-Core besitzt eine eigene Verbindung mit dem Multiport-Memory-Controller (MPMC) und steuert das ICAP-Modul im FPGA an, das die eigentliche Rekonfiguration durchführt. Die dynamisch rekonfigurierbaren Partitionen sind in den beiden Slots (Slot 1 und Slot 2) des Systems enthalten. Diese Slots sind einerseits mit dem PLB-Systembus verbunden und werden andererseits über den DCR[3]-Bus gesteuert. Damit der Prozessor auf den DCR-Bus zugreifen kann, ist zudem eine DCR-Bridge im System notwendig.

Abbildung 4 zeigt den Aufbau eines Slots und die zur Steuerung der Rekonfiguration und zum Datenaustausch nötige Hardware. Bei dem Slot selbst handelt es sich um die im FPGA definierte partiell

[2] CLB = Configurable Logic Block: Dies sind konfigurierbare Basiszellen im FPGA.
[3] DCR = Device Control Register: Ein einfacherer und hardwareschonender Bus zur Steuerung von Hardwareeinheiten.

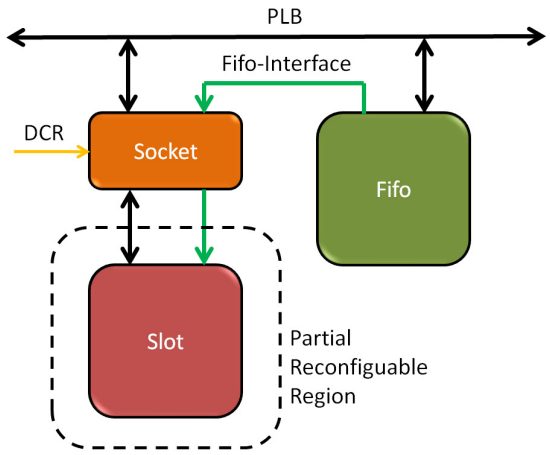


Abbildung 4: Der Aufbau eines Slots: Bei dem Slot handelt es sich um die eigentliche partiell rekonfigurierbare Region im FPGA. Diese ist über die Socket-Bridge mit dem PLB-Bus und FIFO-Interface verbunden. Das FIFO-Modul erlaubt es der CPU mit einem Slot größere Datenmengen, unabhängig vom Rekonfigurationszustand, auszutauschen.

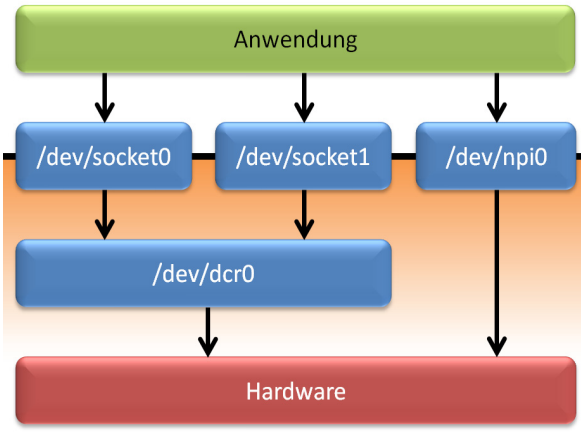


Abbildung 5: Zusammenwirken der zur Rekonfiguration nötigen Treiber: Über den NPI-Treiber kann die Rekonfiguration vorgenommen werden, wogegen es mit den Socket-Treibern möglich ist, die Bus-Verbindung zum Slot ein- und auszuschalten sowie auf den Speicherbereich des Slots zuzugreifen. Der DCR-Treiber wird von den Socket-Treibern benutzt, um die Steuerbefehle an die Socket-Bridges zu übertragen.

rekonfigurierbare Region (Partition). Damit eine dort konfigurierte Hardwareeinheit vom Prozessor angesprochen werden kann, ist dieser mit dem PLB-Systembus verbunden. Während des Rekonfigurationsprozesses treten allerdings auf den Signalen des Slots undefinierte Zustände auf, weswegen der PLB-Systembus nicht direkt an den Slot angeschlossen ist, sondern indirekt über die Socket-Bridge. Diese kann den Systembus vom Slot abtrennen, so dass der Bus während einer Rekonfiguration nicht gestört wird. Die Steuerung der Socket-Bridge geschieht über den DCR-Bus und erlaubt einerseits ein Einschalten und Abschalten des Systembusses und FIFO-Interfaces zum Slot sowie ein Reset desselben. Damit die CPU größere Datenmengen, unabhängig vom Zustand des Slots, austauschen kann, verfügt jeder Slot über ein FIFO-Modul. Dieses enthält einen Eingangs- und einen Ausgangsfifo, die jeweils 512×32 Bit groß sind und auf die der Slot mit Hilfe des FIFO-Interfaces zugreifen kann.

B. Softwareabstraktion

Als Betriebssystem wurde ein Linux mit dem Kernel 2.6 eingesetzt. Ziel des Projekts war es, die Rekonfiguration der Hardware auf Betriebssystemebene zu abstrahieren. Hierzu wurden ausschließlich Kernel-Treibermodule genutzt, wodurch kein Eingriff in den Kernel-Sourcecode nötig war. Außerdem sind alle Teilfunktionen der Rekonfiguration in eigenständigen Treibern realisiert worden, wodurch es möglich war jede Funktion in sich zu kapseln und die Abstraktion stufenweise zu erhöhen. Folgende Abstraktionsstufen wurden in diesem Projekt bearbeitet:

- Die grundlegende Möglichkeit über das Betriebssystem Hardwareeinheiten in den Slots zu rekonfigurieren.
- Eine vom Betriebssystem automatisch eingeleitete Rekonfiguration, falls eine bestimmte Hardwareeinheit benötigt wird (*Virtuelle Hardware*).
- Eine vom Betriebssystem automatisch durchgeführte Rekonfiguration im Kontext einer komplexen Aufgabenstellung, welche das Zusammenspiel mehrerer Hardwareeinheiten benötigt (*Virtuelle Algorithmen*).

C. Rekonfiguration über die Kommandozeile

Um die Rekonfiguration durch das Betriebssystem grundlegend zu ermöglichen, wurden drei Treibermodule implementiert: Der *DCR*-Treiber, welcher die Übertragung von Steuerbefehlen auf dem DCR-Bus ermöglicht, der *Socket*-Bridge Treiber, mit dem eine bestimmte Socket-Bridge ein- und ausgeschaltet werden kann, sowie der *NPI*-Treiber, über den die eigentliche Rekonfiguration vorgenommen wird (siehe Abbildung 5).

Jedes Treibermodul besitzt kernelinterne und -externe Schnittstellen. Über die externen Schnittstellen können Anwendungsprogramme die Funktionen des Treibers nutzen und über interne Schnittstellen stellt der Treiber seine Funktionalitäten anderen Treibern zur Verfügung. Im Folgenden werden die unterstützten Funktionalitäten der drei Treiber erläutert.

1) DCR-Treiber

Der DCR-Treiber ist für das Lesen und Schreiben von DCR-Registern zuständig. Soll auf ein Register geschrieben werden, erwartet der Treiber von der Anwendung oder einem anderen Treiber die Registeradresse sowie das zu schreibende Datum. Beim Lesevorgang benötigt er nur die Registeradresse und gibt das gelesene Datum zurück.

2) Socket-Treiber

Der Socket-Treiber erlaubt das Ein- und Abschalten der Signale eines Slots sowie den Zugriff auf den Speicherbereich des Slots. Weist eine Anwendung oder ein anderer Treiber den Socket-Treiber an den Slot ein- bzw. abzuschalten, so schreibt dieser mit Hilfe des DCR-Treibers die hierzu nötigen Werte in das zur jeweiligen Socket-Bridge zugehörige DCR-Register. Sobald der Socket-Treiber im System aktiviert wird, allokiert er die zum Slot zugehörige Speicheradressen und erlaubt anschließend anderen Treibern oder Anwendungen, die Register innerhalb dieses Speicherbereichs zu beschreiben oder zu lesen. Bei diesem Speicherbereich handelt es sich um den in der Memory-Map des Systems zugeordneten Adressraum des jeweiligen Slots, auf den der Prozessor über den Systembus zugreifen kann. Es ist für andere Treiber alternativ auch möglich die Kontrolle über den Slot-Speicherbereich selbst zu übernehmen, wobei der Socket-Treiber diesem Treiber dazu die physikalische Basisadresse und die Größe des Speicherbereichs übermittelt, so dass dieser selbst den Speicher allokieren kann.

3) NPI-Treiber

Der NPI-Treiber übernimmt die Rekonfiguration des FPGAs. Hierzu müssen ihm über einen anderen Treiber oder von einer Anwendung die zur Konfiguration nötigen partiellen Bitdaten übergeben werden. Der NPI-Treiber überprüft dabei den Header der Bitdaten und speichert anschließend die eigentlichen Konfigurationsdaten in einem DMA-fähigen Speicherbereich ab. Anschließend schreibt er die physikalische Startadresse sowie die Größe der Konfigurationsdaten in die Register des NPI-Hardwaremoduls und startet den Rekonfigurationsvorgang auf Hardwareseite.

Das Zusammenspiel dieser Treiber erlaubt eine einfache Rekonfiguration einzelner Slots, welche auch von der Kommandozeile aus möglich ist. Als erstes wird hierzu mit Hilfe des, dem Projekt beiliegenden, Programms *slotctrl* der zu rekonfigurierende Slot abgeschaltet:

```
# slotctrl  /dev/socket0  disable
```

Anschließend kann dem NPI-Modul die zum Slot 0 passende partielle Bitdatei übergeben werden:

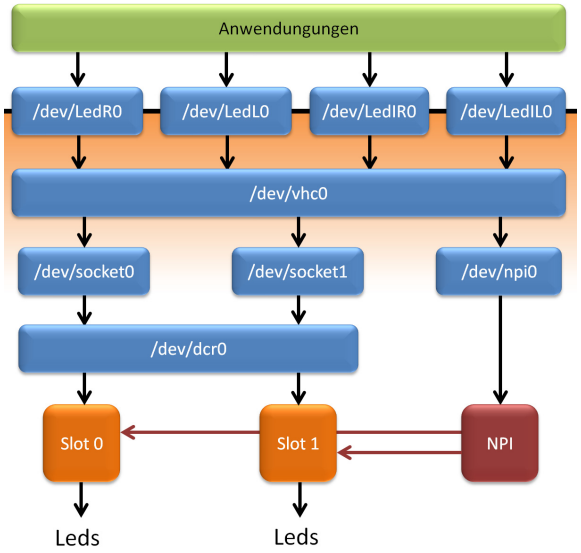


Abbildung 6: Zusammenwirken der Treiber für das Konzept der Virtuellen Hardware: Eine virtuelle Hardwareeinheit besitzt einen eigenen Treiber (HWA0 … HWD0) und meldet einen Zugriff durch die Anwendung an den Virtual Hardware Controller (vhc0) weiter. Dieser übernimmt die Rekonfiguration in einem freien Slot oder gibt eine Fehlermeldung zurück, falls alle Slots belegt sind.

```
# cat  /mnt/cfcard/ledslot0.bit  >  /dev/npi0
```

Nach dem Rekonfigurationsvorgang muss der Slot wieder eingeschaltet werden, damit ein Zugriff auf ihn möglich wird:

```
# slotctrl  /dev/socket0  enable
```

Dieses Vorgehen zur Rekonfiguration ist an den im Artikel [11] erläuterten Ansatz angelehnt worden. Neu ist hierbei das Ein- und Abschalten des entsprechenden Slots, was nötig ist, um den Systembus vom Slot zu trennen, so dass undefinierte Signalzustände während der Rekonfiguration das System nicht zum Absturz bringen. Das NPI-Modul kann dieses Abschalten nicht automatisch übernehmen, da in den Bitdaten kein geeigneter Hinweis auf den zu rekonfigurierenden Slot enthalten ist.

D. Virtuelle Hardware

Die im vorhergehenden Kapitel beschriebene Rekonfiguration ist für Anwendungsprogramme immer noch relativ umständlich und vor allem findet keine Zugriffskontrolle des Betriebssystems bezüglich der Slots statt. Damit kann es passieren, dass mehrere Anwendungen gleichzeitig eine Hardwareeinheit in denselben Slot konfigurieren oder eine soeben konfigurierte Hardware von einer anderen Anwendung wieder gelöscht wird. Um diese konkurrierenden Zugriffe zu vermeiden, ist eine höhere Abstraktion des

Vorgangs notwendig, wobei ausschließlich das Betriebssystem eine Rekonfiguration vornimmt. Dieses entscheidet dann, welche Anwendungen in welcher Reihenfolge die konfigurierten Slots nutzen können. Genau diese Aufgaben übernimmt ein weiteres Treibermodul mit dem Namen *Virtual Hardware Controller* (VHC).

Der *Virtual Hardware Controller* (siehe Abbildung 6) stellt damit die Schnittstelle zwischen dem Hardwaretreiber (der virtuellen Hardwareeinheit) und den zur Rekonfiguration nötigen Treibern dar. Wird ein Treiber einer virtuellen Hardwareeinheit im System angemeldet, so reicht dieser den sogenannten *Virtual Hardware Stream* (VHS) an den VHC weiter. Diese Datei wird aus den partiellen Bitdaten erzeugt und enthält alle Informationen, die vom VHC zur Rekonfiguration benötigt werden. Das Weiterreichen des VHS macht die virtuelle Hardware im System bekannt und der VHC speichert dabei die zur Rekonfiguration benötigten Bitdaten im Hauptspeicher ab. Da die Bitdaten in diesem Moment schon im System vorhanden sind, aber die Hardware physikalisch noch nicht existiert, wird hierbei von *virtueller Hardware* gesprochen. Erst wenn ein Anwendungsprogramm tatsächlich auf den Treiber einer virtuellen Hardware zugreift, gibt dieser ein Signal an den VHC, welcher in einem freien Slot die Rekonfiguration vornimmt. Sind zu diesem Zeitpunkt alle Slots belegt, wird entweder (je nach Aufruf des VHC) eine Fehlermeldung zurückgegeben oder die Anfrage in einer Warteschleife gespeichert.

Nachdem der VHC die Rekonfiguration erfolgreich vorgenommen hat, wird dem Treiber der virtuellen Hardware die Basisadresse und Größe des Speicherbereichs mitgeteilt, welcher dem, in der Memory-Map definierten, Adressraum des jeweiligen Slots entspricht. Der Hardwaretreiber kann diesen Speicherbereich allokieren und anschließend darauf zugreifen, als ob es sich um eine herkömmliche Hardwareeinheit handeln würde. Sobald kein weiterer Zugriff auf die Hardware mehr benötigt wird, gibt der Hardwaretreiber diesen Speicherbereich wieder frei und meldet dem VHC, dass der Slot als frei markiert werden kann. Anschließend wäre es möglich in diesem Slot eine andere virtuelle Hardwareeinheit zu rekonfigurieren.

Eine partielle Bitdatei beschreibt immer die Konfiguration einer dynamischen Hardwareeinheit in einem konkreten Slot. Sind daher zwei Slots im System vorhanden, müssen für eine Hardwareeinheit zwei partielle Bitdateien vorliegen, von der jede jeweils einem Slot zugewiesen ist. Diese partiellen Bitdateien werden mit Hilfe eines Übersetzungsprogramms (*Virtual Hardware Translator – vht*) zu einer einzigen Datei, dem *Virtual Hardware Stream* zusammengefügt. Damit beinhaltet diese Datei alle Informationen, welche der VHC benötigt, um eine Hardwareeinheit in unterschiedlichen Slots zu konfigurieren. Zusätzlich können auch Gewichtungsfaktoren angegeben werden, die dazu führen, dass bestimmte Slots im System vorrangig für eine Hardwareeinheit genutzt werden. Kann dagegen eine Hardwareeinheit in einem Slot gar nicht rekonfiguriert werden (weil beispielsweise der Slot zu klein ist), ist auch dies in der VHS vermerkt, so dass der VHC einen anderen freien Slot auswählt.

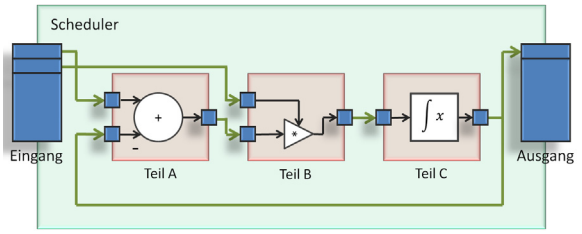


Abbildung 7: Ein möglicher virtueller Algorithmus: Teil A erhält als Eingangsdaten den ersten Eingangswert des Gesamtalgorithmus sowie den letzten Ausgangswert von Teil C. Teil B erhält als Eingangswerte den zweiten Eingangswert des Gesamtalgorithmus sowie den Ausgangswert von Teil A, und Teil C verarbeitet direkt den Ausgangswert von Teil B. Der Ausgangswert von Teil C stellt dabei auch gleichzeitig den Ausgangswert des Gesamtalgorithmus dar.

E. Virtueller Algorithmus

Eine weitere Abstraktionsstufe stellt das Konzept des *virtuellen Algorithmus* dar. Hierbei wird davon ausgegangen, dass eine komplexe Berechnungsaufgabe, die in Form eines Hardware-Algorithmus gegeben ist, sich nicht vollständig im FPGA integrieren lässt, weswegen sie in mehrere, möglichst gleich große Teilalgorithmen aufgeteilt wurde. Die einzelnen Teilalgorithmen liegen dabei als VHS-Datei vor. Ein weiteres Übersetzungsprogramm (der *Virtual Algorithm Translator – vat*) fügt diese VHS-Dateien, mit weiteren Informationen zum Algorithmus, zu einer VAS-Datei (*Virtual Algorithm Stream*) zusammen. Diese VAS-Datei enthält neben den zur Rekonfiguration nötigen Daten auch eine Anleitung, in welcher Weise die Teilalgorithmen Daten miteinander austauschen.

Somit ist es möglich auch Algorithmen darzustellen, bei denen sich die Eingangswerte der Teilalgorithmen nicht zwangsläufig aus den Ausgangswerten der vorhergehenden Teile ergeben (siehe Abbildung 7). Die Berechnung innerhalb der Teilalgorithmen wird von der zugrundeliegenden Hardwareeinheit vorgenommen, während der Datenaustausch vom *Virtual Algorithm Controller* (VAC) durchgeführt wird. Damit der VAC nicht jedes Datum zwischen den Slots einzeln austauschen muss, benutzt dieser die Fifos der Slots. Damit kann er den Slots größere Datenmengen auf einmal zur Verfügung stellen oder lesen, selbst wenn der Slot gerade rekonfiguriert wird. Um die Fifos anzusprechen, wurde neben dem VAC-Treiber noch ein Fifo-Treiber hinzugefügt (siehe Abbildung 8).

Der virtuelle Algorithmus selbst wird gegenüber den Anwendungsprogrammen mit Hilfe eines Algorithmus-Treibers zugänglich gemacht. Dieser gibt bei seiner Initialisierung im System den VAS (*Virtual Algorithm Stream*) an den VAC weiter, welcher die

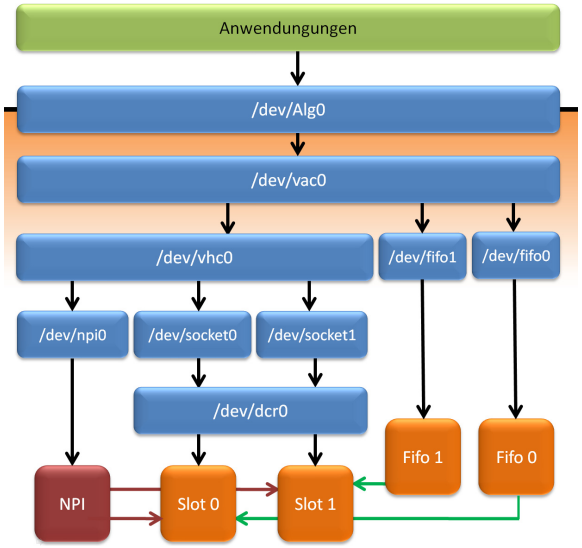


Abbildung 8: Das Zusammenspiel der unterschiedlichen Treiber beim Virtuellen-Algorithmus-Konzept. Der Treiber des Algorithmus reicht die zu berechnenden Daten an den Virtual Algorithm Controller (VAC) weiter, welcher dann mit Hilfe des VHC die Teilalgorithmen abwechselnd in den zur Verfügung stehenden Slots konfiguriert. Die Ein- und Ausgangsdaten der Teilalgorithmen werden mit Hilfe der Slot-Fifos über den Fifo-Treiber geschrieben und gelesen.

dort enthaltenen Daten auswertet, die VHS der Teilalgorithmen an den VHC weiterleitet und somit den Algorithmus im System verfügbar macht. Anschließend kann der Algorithmustreiber die Eingangsdaten an den VAC senden, damit diese berechnet werden. Hierbei sorgt ein Hardware-Scheduler dafür, dass die Teilalgorithmen in der richtigen Reihenfolge in den zur Verfügung stehenden Slots rekonfiguriert werden und so lange dort vorhanden bleiben, bis sie ihre Berechnung ausgeführt haben. Ist ein Teilalgorithmus mit der Berechnung aller Eingangsdaten fertig, wird der Slot wieder freigegeben, so dass dieser für weitere Teilalgorithmen genutzt werden kann. Der Scheduler geht dabei folgendermaßen vor:

1) Alle Teilalgorithmen beim VHC auf die Warteliste setzen.
2) Alle Teilalgorithmen der Reihe nach abarbeiten:
   a) Warten, bis der Teilalgorithmus in einem freien Slot rekonfiguriert wird oder fertig rekonfiguriert ist.
   b) Übertragen der Eingangsdaten in den Eingangsfifo des Slots.
   c) So lange die Ausgangsdaten vom Fifo lesen, bis der Teilalgorithmus seine Berechnung abgeschlossen hat.
3) Den Slot wieder freigeben.
4) Den ersten Teilalgorithmus auf die Warteliste des VHC setzen.
5) Die Ausgangsdaten des Gesamtalgorithmus zusammenstellen und für den Algorithmustreiber abspeichern.

Während der Scheduler auf die erfolgreiche Rekonfiguration oder auf die Ausgangsdaten eines Teilalgorithmus wartet, legt sich der VAC im System schlafen, so dass in der Zwischenzeit andere Prozesse berechnet werden können.

Bevor die Ausgangsdaten an den Algorithmustreiber weitergereicht werden, wird noch einmal der erste Teilalgorithmus auf die Warteliste gesetzt. Dahinter verbirgt sich die Annahme, dass normalerweise mehrere Datensätze auf einmal berechnet werden sollen und sofort nach dem Berechnen eines Datensatzes ein neuer an den VAC übergeben wird. Ist dies der Fall, ist der erste Teilalgorithmus schon erfolgreich konfiguriert und der VAC kann sofort mit dem Übertragen der Daten in den Eingangsfifo beginnen. Wenn der Algorithmustreiber dem VAC meldet, dass die Software den Algorithmus nicht mehr benötigt, werden alle noch verbliebenen Teilalgorithmen aus der Warteliste gelöscht und eventuell schon konfigurierte Slots werden freigegeben.

VI. Fazit

A. Zusammenfassung

Eines der bisherigen Probleme bei der praktischen Anwendung dynamischer, partieller Rekonfiguration, ist der zusätzliche Aufwand bei der Entwicklung eines solchen Systems auf Hardware- und Softwareseite. So ist es beispielweise nötig, weitere Hardwaremodule (wie das NPI- und die FIFO-Module) in das System zu integrieren, Algorithmen sinnvoll aufzuteilen und die partiell rekonfigurierbaren Partitionen im FPGA zu definieren sowie ihnen die nötigen Ressourcen zuzuweisen. Auf der Softwareseite sind Programme nötig, die mit der Rekonfiguration umgehen können, diese einleiten oder überwachen und die Zugriffe auf die Slots zeitlich koordinieren können. Dieser zusätzliche Aufwand ist vermutlich mit daran schuld, dass dynamisch rekonfigurierbare Rechensysteme derzeit nur selten in praktischen Anwendungen eingesetzt werden.

Der vorliegende Beitrag versuchte hierzu zwei Möglichkeiten aufzuzeigen, den Aufwand zumindest auf der Softwareseite des Systems zu reduzieren. Mit Hilfe des *Virtual Hardware Controllers* und des *Virtual Algorithm Controllers* ist es möglich, Anwendungen zu entwickeln, welche von den Vorteilen der Rekonfiguration profitieren, ohne dass sie sich um den komplexen Vorgang derselben und des Datenaustauschs mit den Slots kümmern müssen. Für die Anwendungen spielt es tatsächlich gar keine Rolle mehr, ob die benutzte Hardware dauerhaft oder nur zeitweise im System vorhanden ist. Selbst den Programmierern der Hardware- und Algorithmustreiber wird die mit

der Rekonfiguration verbundene Arbeit fast vollständig abgenommen. Mit einfachen Befehlen lässt sich sowohl der VHC als auch der VAC über eine kernelinterne Schnittstelle ansprechen, so dass sich ein herkömmlicher Linux-Treiber leicht um die Möglichkeit der dynamischen Rekonfiguration von Hardware erweitern lässt.

Über diese Erleichterung der Rekonfiguration hinaus bieten die vorgestellten Lösungen auch weitere Vorteile gegenüber statischer und damit nicht rekonfigurierbarer Hardware: Mit Hilfe des VHCs ist es möglich, ein und dieselbe Hardwareeinheit so oft im gleichen System zu rekonfigurieren, wie Slots vorhanden sind. Greifen also mehrere Anwendungen gleichzeitig auf eine Hardwareeinheit zu, kann diejenige Hardwareeinheit einfach mehrfach im System zur Verfügung gestellt werden. Der VAS hingegen erlaubt das Abarbeiten von sehr komplexen Aufgabenstellungen mit nur wenigen Slots. Zwar vergrößert sich die zur Berechnung nötige Zeit, je mehr Algorithmusteile und je weniger Slots vorhanden sind, doch dies ist oft weniger kritisch als die höheren Kosten, die mit der Auswahl eines größeren FPGAs verbunden wären. Darüber hinaus passt sich der VAC automatisch der jeweiligen Situation im System an: Ist während einer Berechnung plötzlich ein Slot nicht mehr verfügbar (da beispielsweise der VHC gerade eine virtuelle Hardware konfiguriert hat) wird die Berechnung mit den restlichen zur Verfügung stehenden Slots weitergeführt.

B. Ausblick

Obwohl es sich bei dem hier beschriebenen Projekt um eine Bachelorthesis handelt und der zeitliche Rahmen somit beschränkt war, ist das Thema der dynamisch partiellen Rekonfiguration an der Hochschule Pforzheim weiterhin ein wichtiger Bestandteil der praktischen Forschungstätigkeit. Somit ist davon auszugehen, dass in zukünftigen Projekt-, Forschungs- und Abschlussarbeiten auch weiterhin auf Grundlage der bisherigen Arbeit die praktische Realisierbarkeit von Anwendungen mit dynamisch rekonfigurierbaren Rechensystemen untersucht und verbessert wird.

Eine wichtige Erweiterung des beschriebenen Projektes wären hierbei direkte Datenpfade zwischen den Fifos der einzelnen Slots. Damit wäre es möglich, den Hardware-Scheduler fast vollständig von der Aufgabe des Umkopierens der Daten von einem Fifo zu einem anderen zu entlasten. Der Prozessor könnte sich dann innerhalb dieser Zeit um andere Aufgaben kümmern, was die Performance solcher Systeme weiter steigern würde. Weiterhin wäre es denkbar, den Hardware-Scheduler komplett in Hardware zu realisieren oder den Scheduling-Vorgang der Softwarelösung weiter zu optimieren. Der Autor der zugrundeliegenden Bachelorthesis hofft, mit seiner Arbeit eine solide Grundlage für diese und weitere zukünftige Entwicklungen auf dem Gebiet dynamisch rekonfigurierbarer Rechensysteme gelegt zu haben.

LITERATURVERZEICHNIS

[1] M. Scheffler, F. Kesel: „Dynamisch partielle Rekonfiguration“, *Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg*, Reutingen, Juli 2010.

[2] M. Platzner, J. Teich, N. Wehn (Hg.): „Dynamically Reconfiguable Systems. Architectures, Design Methos and Applications”, *Springer*, Dordrecht 2010.

[3] S. Hauck, A. Dehon: „Reconfigurable Computing. The Theory and Practice of FPGA-based Computation”, *Morgan Kaufmann,* Burlington 2008.

[4] S. C. Goldstein, H. Schmit, M. Moe, M. B. S. Cadambi, R. R. Taylor, R. Laufer: „PipeRench: A Coprocessor for Streaming Multimedia Acceleration”, *In Proceeding of the 26th International Symposium on Computer Architecture,* May 1999.

[5] S. Raaijmakers, S. Wong: „Run-time partial reconfiguration for removal, placement and routing on the Virtex-II Pro”, *In Proceeding of the Field Programmable Logic and Applications International Conference,* August 2007.

[6] A. Megacz: „A Library and Platform for FPGA Bitstream Manipulation”, *In Proceeding of the 15th Annual IEEE International Symposium on Field-Programmable Custom Computing Machines*, April 2007.

[7] A. Jara-Berrocal, A. Gorden-Ross: „VAPRES: A Virtual Architecture for Partially Reconfigurable Embedded Systems”, *In Proceeding of the Design, Automation & Test in Europe Conference & Exhibition (DATE),* March 2010.

[8] D. Gajski, R. Kuhn: „Guest Editor’s Introduction: New VLSI Tools”, *IEEE Computer,* December 1983.

[9] F. Kesel, R. Bartholomä: „Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs“, *Oldenbourg,* München 2009[2].

[10] I. Richardson, M. Bystrom, S. Kannangare, M. de Frutos Lòpez: „Dynamic Configuration: Beyond video coding standards”, *In Proceeding of the 21th International System on Chip Conference,* September 2008.

[11] J. Williams, N. Bergmann: „Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip”, *In Proceeding of the International Conference on Engineering of Reconfigurable Systems and Algorithms*, June 2004.

HOCHSCHULE
KONSTANZ
TECHNIK, WIRTSCHAFT
UND GESTALTUNG

HART-Transmodulator mit DDS auf einem FPGA

Albert Schaaf, Irenäus Schoppa

***Zusammenfassung*—Das HART-Kommunikationsprotokoll ist ein weit verbreiter Übertragungsstandard in der Prozessautomatisierung. Er beruht auf der Überlagerung von CPFSK-modulierten digitalen Daten auf einem analogen Stromsignal und ermöglicht somit einen digitalen Datenaustausch zwischen Feldgeräten. Im Rahmen eines Pilotprojektes wurde die Bitübertragungsschicht (physical layer) des HART-Protokolls auf einem Spartan-3 implementiert. Der entwickelte HART-Transmodulator verfügt einerseits über eine UART-Schnittstelle mit einer in Hardware implementierten Datenflusssteuerung mittels RTS/CTS-Handshaking. Andererseits ist er mit einer SPI-Schnittstelle für den Anschluss an einen Analog-Digital-Wandler und einen Digital-Analog-Wandler ausgestattet. Die Umsetzung der digitalen Informationen auf das analoge Stromsignal erfolgt nach dem Bell-202-Standard. Die CPFSK-Modulation mit kontinuierlicher Phase wurde auf der Grundlage der direkten digitalen Signalsynthese realisiert. Die CPFSK-Demodulation erfolgt mittels eines Nulldurchgangdiskriminators, der Nulldurchgänge des mit zwei Signalfrequenzen codierten CPFSK-Signals detektiert. Die Messung des zeitlichen Abstandes zwischen den Nulldurchgängen ermöglicht es, die Signalfrequenzen als digitale Daten zu interpretieren.**

***Schlüsselwörter*—HART-Modem, FPGA, CPFSK-Modulation und Demodulation, Digitale Signalsynthese.**

I. Einleitung

HART (Highway Addressable Remote Transducer) ist ein in der Prozess- und Automatisierungstechnik weit verbreitetes Kommunikationsprotokoll, dessen Einsatz durch die herstellerunabhängige und nicht kommerzielle Organisation HCF (HART Communication Foundation) gefördert wird. Die HCF sichert auch die Aufrechterhaltung des offenen Protokollstandards, koordiniert dessen Weiterentwicklung und leistet vielseitige technische Unterstützung. Das gesamte HART-Protokoll ist in einer umfangreichen HCF-Spezifikation detailliert dokumentiert. Für das HART-Protokoll sind in der Spezifikation nach dem ISO-/OSI-Schichtenmodell drei Schichten definiert: die Bitübertragungsschicht (physical layer), die Datensicherungsschicht (data link layer) und die Anwendungsschicht (application layer). Bei der Durchführung dieses Projektes waren vor allem die Quellen [1] und [2] mit der Beschreibung der Bitübertragungsschicht relevant.

A. Schaaf, alschaaf@htwg-konstanz.de, ist Student an der HTWG-Konstanz, I. Schoppa, ischoppa@htwg-konstanz.de, ist Mitglied der HTWG Konstanz, Brauneggerstr. 55, 78462 Konstanz.

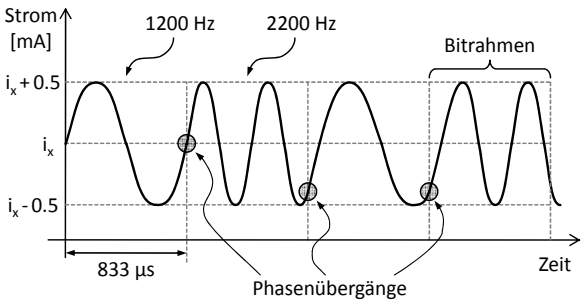


Abbildung 1: Prinzip der phasenkontinuierlichen Frequenzmodulation des Bitmusters 1010.

Das HART-Kommunikationsprotokoll verdankt seine weltweite Akzeptanz und Verbreitung u.a. der Tatsache, dass Daten zwischen HART-konformen Geräten physikalisch auf der konventionellen, analogen 4–20 mA-Stromschleife übertragen werden. Diese Übertragung erfolgt mittels der phasenkontinuierlichen Frequenzmodulation (Continuous Phase Frequency Shift Keying, CPFSK) nach dem Bell 202-Standard mit einer festen Bitrate von 1200 bps. Eine logische ’0’ wird durch die Frequenz 2200 Hz und eine logische ’1’ durch die Frequenz 1200 Hz repräsentiert. Die Abbildung 1 zeigt an einem Beispiel den Verlauf eines frequenzmodulierten digitalen Signals für das Bitmuster 1010. Das modulierte HART-Signal ist mit einer Amplitude von ± 0,5 mA symmetrisch und wird einem analogen (Mess-)Signal überlagert. Das mittelwertfreie HART-Signal beeinflusst eine zeitgleich stattfindende Übertragung des analogen (Mess-)Signals nicht. Neben der Signalübertragung dient die Stromschleife bei der 2-Leiter-Technik auch als Versorgung der Feldgeräte.

Das HART-Protokoll beruht auf dem Master-Slave-Prinzip und ermöglicht den Datenaustausch zwischen

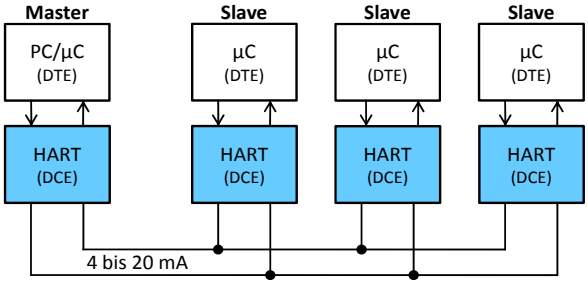


Abbildung 2: Netzwerkkonfiguration eines typischen (Meß-)Systems im Multi-Drop-Modus mit HART-fähigen Komponenten.

einer Leitstation (Master) und einem oder mehreren Feldgeräten (Slaves), z.B. Messwertumformer, Sensoren, Steuerungen oder Regler. Neben einer Leitstation, die normalerweise als primärer Master im System ausgewiesen ist, unterstützt das HART-Protokoll auch einen sekundären Master, z.B. Handterminal. Als Netzwerkkonfigurationen sind Punkt-zu-Punkt-Verbindungen (mit bis zu zwei Mastern) oder Multi-Drop-Verbindungen mit bis zu 15 Feldgeräten möglich. In der Abbildung 2 ist eine typische Netzwerkkonfiguration im Multi-Drop-Modus mit einem Master und drei Slaves dargestellt. Jede Komponente ist mit einer HART-Schnittstelle für den Anschluss an die 4–20 mA-Stromschleife ausgestattet.

Die HART-Datenübertragung erfolgt bidirektional im Halbduplex-Modus und basiert auf der asynchron-seriellen Bitübertragung mit der festen Baudrate von 1200 bps, mit einem Startbit, acht Datenbits, einem Paritätsbit und einem Stoppbit. Die elf Bits bilden zusammen ein sog. HART-Zeichen, aus denen dann HART-Telegramme zusammengesetzt sind [2]. Nach jeder Übertragung eines HART-Telegramms wird das FSK-Trägersignal abgeschaltet, damit die entsprechende Gegenstelle ihrerseits Daten übertragen kann.

Für die Realisierung der Bitübertragungsschicht des HART-Protokolls sind gegenwärtig zwei Lösungsansätze weit verbreitet:

1. die Hardware-Realisierung mit diskreten, spezialisierten HART-Modems z.B. A5191HRT [3], DS8500 [4] oder HT2012 [5],
2. die Software-Realisierung auf der Basis eines universellen Mikrocontrollers z.B. MSP430 [6] oder eines DSP-Prozessors z.B. [7].

Der in diesem Beitrag beschriebene Ansatz geht von einer dritten Realisierungsmöglichkeit aus, und zwar der Realisierung der Bitübertragungsschicht mit einem FPGA-Baustein. Die Leistung und die Komplexität moderner FPGA-Bausteine ermöglichen uns heute, auf einem Baustein ein vollständiges System zu integrieren, das aus einem Soft-Core-Prozessor, einem Speicher, mehreren Ein-/Ausgabeschnittstellen wie UART oder SPI sowie aus applikationsspezifischen Komponenten wie z.B. einem Kommunikationscontroller oder einem HART-Transmodulator besteht.

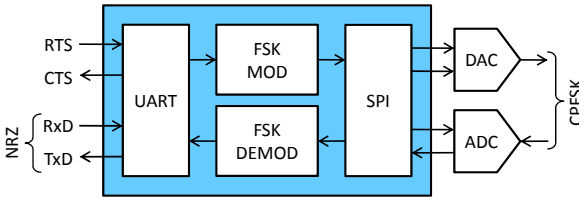


Abbildung 3: Blockschaltbild des HART-Transmodulators mit DA- und AD-Wandlern.

Tabelle 1: Schnittstellensignale des HART-Transmodulators

Signal	Port	Beschreibung
rxd	in	Serielles Datensignal
txd	out	Serielles Datensignal
rts	in	Request To Send
cts	out	Clear to Send
sclk	out	Serial Clock
mosi	out	Master-Output-Slave-Input
miso	in	Master-Input-Slave-Output
csadc	out	Chip-Select für den AD-Wandler
csdac	out	Chip-Select für den DA-Wandler
reset	in	Globales Rücksetzsignal
clock	in	Zentrales Taktsignal
sel	in	Auswahl der Baudrate

II. HART-Transmodulator

Mit dem Begriff Transmodulation bezeichnet man die Umsetzung von Signalen einer Modulationsart in Signale einer anderen Modulationsart, und ein Transmodultor ist eine Vorrichtung zur Durchführung dieser Umsetzung. Bei dem hier vorgestellten HART-Transmodulator handelt es sich um die Umsetzung digitaler, NRZ-modulierter Signale in analoge, CPFSK-modulierte Signale. In der Abbildung 3 ist ein Blockschaltbild des hier entwickelten HART-Transmodulators mit dessen Schnittstellen zu sehen. Im Wesentlichen besteht er aus einem UART-Modul mit einer asynchron-seriellen Schnittstelle für den Anschluss an einen PC/Mikrocontroller, einem SPI-Modul mit der synchron-seriellen Schnittstelle für den Anschluss an AD- und DA-Wandler, einem FSK-Modulator und einem FSK-Demodulator. Weitere Module, wie z.B. ein Baudratengenerator oder zusätzliche Taktteiler, sind hier der Übersichtlichkeit halber nicht dargestellt.

Alle Signale aus der Schnittstelle des HART-Transmodulators, die in der Tabelle 1 zusammengefasst sind, lassen sich in drei Gruppen aufteilen:

- zwei Daten- und zwei Handshake-Signale für die asynchron-serielle Datenübertragung mit einem PC/Mikrocontroller,
- fünf SPI-Signale zur Ansteuerung und Datenübertragung von und zu AD-/DA-Wandlern,
- sonstige Signale wie Takt, Reset oder Auswahl der Baudrate.

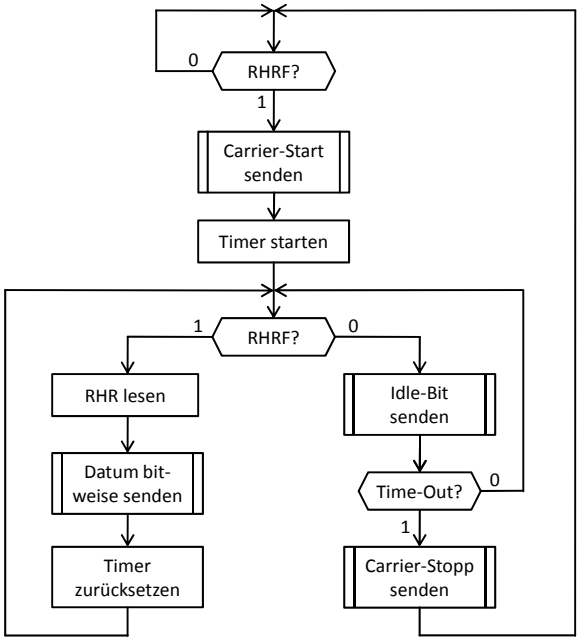


Abbildung 4: Ablaufsteuerung im CPFSK-Modulator.

Die Kommunikationsschnittstelle zwischen einem PC/Mikrocontroller (DTE - Data Terminal Equipment) und dem HART-Transmodulator (DCE - Data Communication Equipment) wurde in Anlehnung an [8] und [9] realisiert. Sie sieht neben zwei Signalen zur Datenübertragung RxD und TxD auch zwei Handshake-Signale RTS (Request To Send) und CTS (Clear To Send) vor. Weil im DCE keine internen Maßnahmen zur Datenpufferung vorhanden sind, und weil die Datenverarbeitung im DCE und im DTE meistens mit unterschiedlichen Geschwindigkeiten stattfinden kann, ist es notwendig, eine automatische Datenflusssteuerung zwischen DCE und DTE mittels zweier Handshake-Signale zu realisieren.

Wenn das DCE bereit ist, seinerseits Daten zu senden, meldet es den Wunsch dem DTE, indem es seinerseits das RTS-Signal aktiviert. Anschließend wartet es so lange, bis das DTE seine Empfangsbereitschaft mit dem CTS-Signal bestätigt hat. Ist das DTE aus irgendeinem Grund nicht mehr in der Lage, die vom DCE ankommenden Daten schnell genug entgegenzunehmen und sie weiterzuverarbeiten, dann deaktiviert das DTE seinerseits das CTS-Signal und kann auf diese Weise den Datentransfer vom DTE drosseln. Daraufhin unterbricht das DTE den Datentransport solange, bis CTS wieder vom DCE aktiviert wird.

Weil die HART-Übertragung bidirektional im Halbduplex-Modus stattfindet, ist für den Sender und Empfänger im HART-Transmodulator eine Steuerung notwendig, die nach einem Sendevorgang auf den Empfangsmodus umschaltet. Der Ablauf dieser Steuerung ist in Abbildung 4 dargestellt. Diese Steuerung hat u.a. die Aufgabe, das FSK-Trägersignal vor der Übertragung eines HART-Telegramms einzuschalten

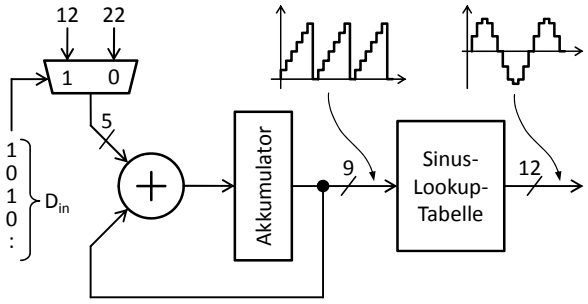


Abbildung 5: Blockschaltbild eines numerisch gesteuerten Oszillators mit einem Phasen-Akkumulator und einer Sinus-Lookup-Tabelle.

(Carrier-Start) und dieses Trägersignal nach der Übertragung wieder abzubauen (Carrier-Stopp).

III. FSK-Modulation

Die Umsetzung eines digitalen Eingangssignals in ein frequenzmoduliertes Ausgangssignal lässt sich mit Hilfe eines numerisch gesteuerten Oszillators NCO (Numerically Controlled Oscillator) realisieren. Eine umfassende Beschreibung dazu findet man z.B. in [10] und [11]. Die Basisschaltung eines NCO ist in der Abbildung 5 zu sehen. Sie besteht aus einer Sinus-Lookup-Tabelle, einem (Phasen-)Akkumulator, einem Addierer und einem Multiplexer.

Das Herzstück des numerisch gesteuerten Oszillators im HART-Transmodulator ist die Sinus-Lookup-Tabelle mit insgesamt 264 Stützpunkten (quantisierten Amplitudenwerten) einer Sinus-Schwingung mit einer Auflösung von 12 Bits. Die implementierte Auflösung resultiert aus der Auflösung des eingesetzten DA-Wandler TLV5616.

Das digitale Eingangssignal D_{in}, das aus einer Folge von Nullen und Einsen besteht, steuert einen 1-aus-2-Multiplexer an, mit dessen Hilfe die Schrittweite W für den Akkumulationsvorgang bestimmt wird:

$$ACC(n) = (ACC(n-1) + W) \bmod K \qquad (1)$$

$ACC(n)$ ist der Wert des Phasen-Akkumulators zum Takt n, W die Schrittweite und K die Anzahl der Stützpunkte.

Die Schrittweite W beträgt 12, wenn eine Eins als Sinus mit der Frequenz 1200 Hz ausgegeben werden soll oder 22, wenn eine Null als Sinus mit der Frequenz 2200 Hz generiert werden soll.

$$W = \begin{cases} 12 \text{ wenn } D_{in} = \text{'1'} \\ 22 \text{ wenn } D_{in} = \text{'0'} \end{cases} \qquad (2)$$

Die so gewählte Schrittweite wird zum Wert des (Phasen-)Akkumulators addiert, und das Resultat dient dann als Adresse zur Ansteuerung der Sinus-Lookup-Tabelle, die mit einem synchronen RAM-Block des

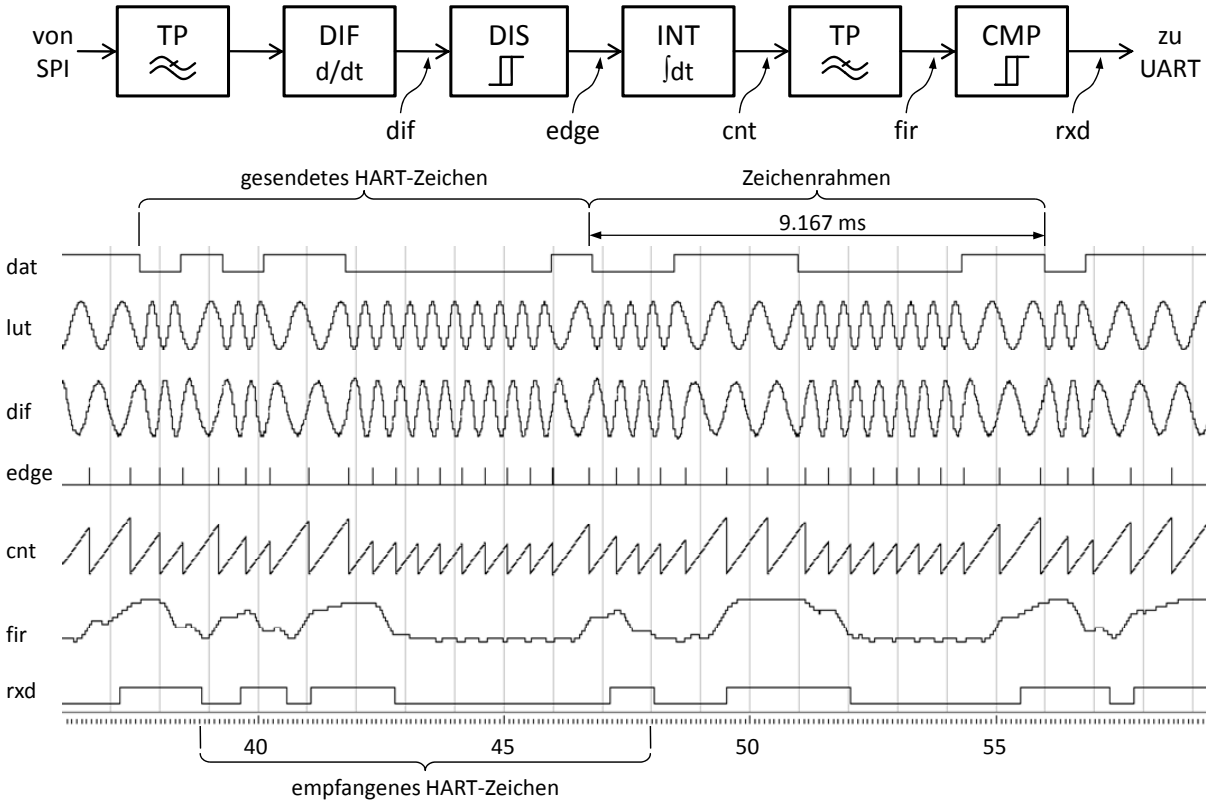


Abbildung 6: Blockschaltbild des FSK-Demodulators und Impulspläne für die Übertragung von zwei HART-Zeichen mit den Datenbytes 0x0D und 0x0E.

FPGAs [12] realisiert wurde. Die Ausgangsfrequenz f_{out} lässt sich mit folgender Formel berechnen:

$$f_{out} = (W \cdot f_{REF}) / K \qquad (3)$$

wobei W die Schrittweite, f_{REF} die Referenzfrequenz und K die Anzahl der Stützpunkte sind. Je größer die Schrittweite, umso höher ist auch die Frequenz der ausgelesenen Sinus-Schwingung. Der numerisch gesteuerte Oszillators NCO im HART-Transmodulator wird mit einer Referenzfrequenz f_{REF} = 26,331 kHz getaktet, was bei den gewählten Schrittweiten und der definierten Anzahl der Stützpunkte die Ausgangsfrequenzen f_{out}('1') = 1197 Hz und f_{out}('0') = 2194 Hz ergibt. Beide Frequenzen erfüllen mühelos die in [1] definierten Toleranzen von ±1%.

IV. FSK-Demodulation

In der Literatur wie z.B. [10], [13] und [14] werden zahlreiche Verfahren zur FSK-Demodulation genannt:

- Flankendiskriminator
- Nulldurchgangsdiskriminator
- Differenzdemodulator
- PLL-Demodulator
- Bandpassfilterung
- Digitale/Schnelle Fourier-Transformation
- Goertzel-Algorithmus

Die meisten dieser Demodulationsverfahren sind mit einem relativ hohen Anteil an Multiplikationen rechenintensiv und beruhen im Wesentlichen auf der Arithmetik komplexer Zahlen. Der FSK-Demodulator des HART-Transmodulators ist eine Kombination aus der Flankenerkennung und der Zeitmessung zwischen Nulldurchgängen. Dieses Verfahren zeichnet sich durch eine leichte Implementierbarkeit aus und kommt – im Vergleich zu anderen Demodulationsverfahren – ganz ohne Multiplikationen und ohne Arithmetik komplexer Zahlen aus. Das Verfahren setzt allerdings voraus, dass das Eingangssignal mit einer höheren Abtastrate erfasst werden muss. Die Abtastfrequenz beträgt 57,6 kHz und ist für eine präzise Zeitmessung zwischen den Nulldurchgängen notwendig. Der Signalverarbeitungspfad, der als Blockschaltbild in Abbildung 6 zu sehen ist, setzt sich aus folgenden Komponenten zusammen: einem Sinc-Tiefpassfilter TP, einem Differenzierer DIF, einer Nulldurchgangserkennung DIS, einem Integrator INT, einem zweiten Sinc^2-Tiefpassfilter TP und einem Vergleicher CMP.

In der Abbildung 6 sind auch einige Timing-Diagramme zu sehen, die ausschnittsweise zeitliche Abläufe von sieben repräsentativen Signalen im Sende- und Empfangspfad des HART-Transmodulators darstellen. Der Verlauf des digitalen Signals *dat* zeigt den seriellen Bitstrom für den NCO nach der parallel-seriellen Umsetzung der Datenbytes 0x0D und 0x0E,

inkl. einem Startbit, einem Parity-Bit und einem Stoppbit. Der Verlauf des Signals *lut* zeigt das frequenzmodulierte Signal *dat* aus dem NCO. Die weiteren Übertragungsstufen wie das Senden und das Empfangen mittels SPI, oder die digital-analoge und die analog-digitale Wandlung sind hier nicht dargestellt. Als weiteres Signal sieht man das Signal *dif* nach der Abtastung, der Filterung mit einem Sinc-Tiefpassfilter und nach der Differenzierung. Das Signal *edge* zeigt einzelne Impulse, die nach der Detektion der Nulldurchgänge generiert worden sind. Dieses Signal dient dazu, einen freilaufenden Zähler periodisch zurückzusetzen. Der momentane Wert des Zählers, der dem zeitlichen Abstand zwischen zwei Nulldurchgängen entspricht, ist als Signal *cnt* zu sehen. Anschließend erfolgt eine weitere Filterung des Signals *cnt* mit einem $Sinc^2$-Tiefpassfilter. Das Resultat nach diesem Schritt ist durch das Signal *fir* dargestellt. Aus diesem Signal wird das digitale Signal *rxd* durch einen Vergleich mit Schwellwerten rekonstruiert.

V. Evaluierungsumgebung

Der HART-Transmodulator wurde modular und hierarchisch in VHDL mit generischen Komponenten synthesegerecht modelliert, mit dem ModelSim-Simulator in verschiedenen Testszenarien umfassend simuliert und anschließend mit Hilfe der ISE-Entwicklungsumgebung 12.3 von Xilinx für den FPGA-Baustein XC3S200-4FTG256C synthetisiert. Für den funktionalen Test des hier entwickelten HART-Transmodulators unter möglichst realen Bedingungen wurde das FPGA-Board [15] über eine seiner Erweiterungsleisten (expansion connectors) um eine analoge Experimentierplatine erweitert. Diese Experimentierplatine beinhaltet u.a. einen AD-Wandler ADS7822, einen DA-Wandler TLV5616 sowie zusätzliche, notwendige Analogbauteile. Als Gegenstück zur Kommunikation mit dem HART-Transmodulator dient eine Evaluierungsplatine [16] mit dem kommerziellen HART-Modulator DS8500.

VI. Ausblick

In einem nachfolgenden Projekt ist es geplant, die SPI-Schnittstelle mit den AD-/DA-Wandlern durch einen Delta-Sigma-Modulator und -Demodulator zu ersetzen, dann die direkte Signalsynthese im FSK-Modulator auf der Basis eines RAM-Blocks und des Phasen-Akkumulators durch ein optimiertes Modul zur Berechnung der abschnittsweise linearisierten Sinus-Funktion zu ersetzen und schließlich die gesamte VHDL-Beschreibung des HART-Transmodulators auf eine Low-Power-FPGA-Technologie zu portieren.

Literaturverzeichnis

[1] HCF: „FSK Physical Layer Specification“, HCF_SPEC-54, Rev. 8.1, *HART Communication Foundation*, 1999.

[2] HCF: „Token-Passing Data Link Layer-Specification“, HCF_SPEC-81, Rev. 8.2, *HART Communication Foundation*, 2007.

[3] AMI: „A5191HRT - HART Modem“, Data Sheet A5191HRT/D, Rev. 4, *Semiconductor Components Industries, Inc.*, 2010.

[4] Maxim: „DS8500 - HART Modem“, Data Sheet, Rev. 1, *Maxim Integrated Products, Inc.*, 2009.

[5] Smar: „HT2012 - HART Modem“, Data Sheet HT2012DS, *Smar Research Corp*., 2003.

[6] TI: „FSK Modulation and Demodulation With the MSP430 Microcontroller, Application Note SLAA037, *Texas Instruments, Inc.,* 1998.

[7] TI: „Implementation of an FSK Modem Using the TMS320C17“, Application Report SPRA080, *Texas Instruments, Inc.,* 1997.

[8] HCF: „High Speed HART Prototype Modem Development - Theory of Operation“, *HART Communication Foundation*, 1998.

[9] DIN 66020: „Funktionelle Anforderungen an die Schnittstellen zwischen Datenendeinrichtungen und Datenübertragungseinrichtungen“, Teil 1 und 2, *Beuth Verlag GmbH*, 1999.

[10] B.-G. Goldberg: „Digital Frequency Synthesis Demystified“, *LLH Technology Publishing*, 1999.

[11] Analog Devices: „A Technical Tutorial on Digital Signal Synthesis“, *Analog Devices, Inc*., 1999.

[12] Xilinx: „Using Block RAM in Spartan-3 Generation FPGAs“, Application Note 463, *Xilinx, Inc*., 2005.

[13] D. Rudolph: „Demodulation frequenzmodulierter Signale“, in WissenHeute - Fachzeitschrift für IT/TK, Wirtschaft und Kommunikation 50 (2004), Nr 4, S. 206-218.

[14] H. Weidenfeller und H. Vlcek: „Digitale Modulationsverfahren mit Sinusträger: Anwendung in der Funktechnik“, *Springer Verlag*, 1996.

[15] Xilinx: „Spartan-3 FPGA Starter Kit Board User Guide“, UG130, Ver. 1.2, *Xilinx, Inc*., 2008.

[16] Maxim: „DS8500 Evaluation Kit“, Data Sheet 19-5641, Rev. 0, *Maxim Integrated Products, Inc.*, 2010.

Albert Schaaf studiert Technische Informatik an der HTWG Konstanz im 7. Semester und befasste sich im Rahmen eines Projektes mit der Implementierung des HART-Transmodulators auf einem Spartan-3 FPGA-Baustein.

Irenäus Schoppa studierte Informatik an der Technischen Universität Berlin und erhielt dort im Jahre 1993 den akademischen Grad Dipl.-Informatiker. Im Jahre 1998 promovierte er dort zum Dr.-Ing.. Seit dem Jahr 2008 ist er Professor für Hardware-Software Codesign an der HTWG Konstanz.

FPGA Based Image Processing Platform: Concept, Design and Implementation of Algorithms as Pipelined Dedicated Hardware Blocks

Simran Singh, Heinz-Peter Bürkle

***Abstract*—Image Processing in Embedded Systems is gaining increasing importance and is a computationally intensive operation on large amounts of data. In this paper Image Processing and Pattern Recognition algorithms development and their performance evaluation in an FPGA as Dedicated Hardware blocks are presented.**

***Index Terms*—FPGA, Image Processing, Circular Hough Transformation, Parallel Computing.**

I. Introduction

Image Processing is becoming increasingly important in embedded systems. It requires high data processing speed due to the large amount of data in each image frame. Processing time becomes even more critical, if real-time constraints are required in the respective application.

Therefore, up until recent years only high speed microprocessors were able to fulfil this highly demanding computational task. Since recent years Field Programmable Gate Arrays (FPGAs) with their increasing computational performance are becoming more interesting for Image Processing tasks.

Usage of FPGAs opens new possibilities of exploiting parallelism to implement Image Processing algorithms. By implementing these algorithms as dedicated hardware blocks, individual blocks can process data in parallel, resulting in enhanced processing speed.

In this paper, an FPGA based Image Processing platform is designed and various options of computational parallelism in an FPGA for Pattern Recognition algorithms are examined. The main differences from a software oriented approach are discussed.

Moreover, the associated development challenges are studied, as well as the resource restrictions. Synthesis and hardware test results of exemplary algorithms are presented.

Figure 1: Scenario for image processing in an FPGA: Circle Detection Algorithm for some coins

II. Image Processing Hardware Blocks

Microprocessor based Image Processing has the advantage of many Open Source software libraries being available. The Image Processing & Pattern Recognition functions implemented in these libraries are offering more than satisfactory results. One well-known example is OpenCV [1]-[2], an open source computer vision library with a wide range of image processing algorithm functions.

On the other hand Open Source hardware cores in Verilog and VHDL are still quite limited. These hardware cores are often not optimally tailored for the respective application and lead to inefficiency in hardware resource usage.

Due to this fact, dedicated hardware blocks in FPGAs often have to be developed for the specific application. Therefore, during development planning it is often necessary to estimate HDL code development effort for a specific application need.

Figure 1 above shows an example Image Processing and Pattern Recognition scenario to be processed in an FPGA. Using a circle detection scenario as a target application, the computational complexity and demand can be made clear, since circle detection algorithms are well known to require much calculation.

Simran Singh, Heinz-Peter Bürkle, Email: heinz-peter.buerkle@htw-aalen.de, Adresse: Beethovenstr. 1, D-73430 Aalen, Germany

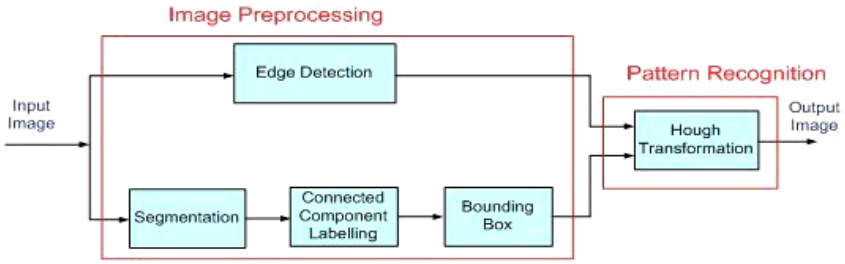


Figure 2: Algorithm Hardware Blocks

Also, before the circle detection algorithm can be executed, the image has to be pre-processed using Image Processing algorithms in order to prepare the image as well as to extract some necessary information from the image, which will be useful in later stages of the image processing.

Therefore, diligence has to be invested into the algorithms to be implemented. This is true for both pre-processing and target algorithms. Each algorithm has to be developed as a hardware block, which will fulfil the defined task. FPGA resource consumption and development time are dependent on the right decisions at this early stage in the development process.

Figure 2 above shows an example with Image Processing hardware blocks required for a Pattern Recognition task. The input image is processed by each block using the pipeline principle.

The advantage of an FPGA implementation is the simultaneous processing of data in each of these blocks. The overall computation speed is increased via exploitation of parallel computing.

For processing purposes the input image is converted into a greyscale image before being fed into the chain of algorithms.

A. Segmentation

Segmentation [3] in our case is a simple straight forward operation that functions as a comparator setting each pixel value to a maximum or minimum value based on a reference value. The goal is to separate an object from the background.

B. Edge Detection

Edge Detection [3] is used to find the edges in an image by determining the changes in intensity in a target region. It is a region based operator that requires information of the neighbouring pixels for calculation of the intensity change at a pixel. In this particular edge detection algorithm the sobel operator is used, which has the convolution masks as given Table 1 and Table 2.

The change in intensity is determined in both x and y direction with the respective convolution mask. The magnitude of change is calculated based on the change in both directions. Image intensity changes suddenly at the position, where an edge is located in contrast to a slow change at other locations.

Table 2: Horizontal Mask

-1	0	1
-2	0	2
-1	0	1

Table 1: Vertical Mask

-1	-2	-1
0	0	0
1	2	1

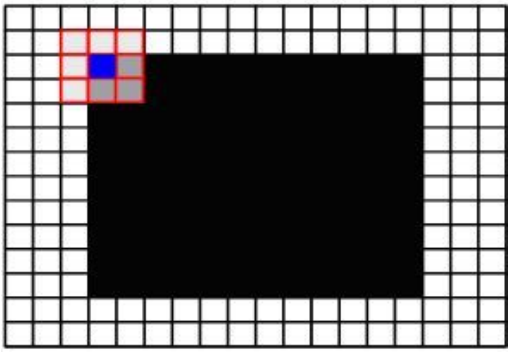


Figure 3: Principle of edge detection convolution operation

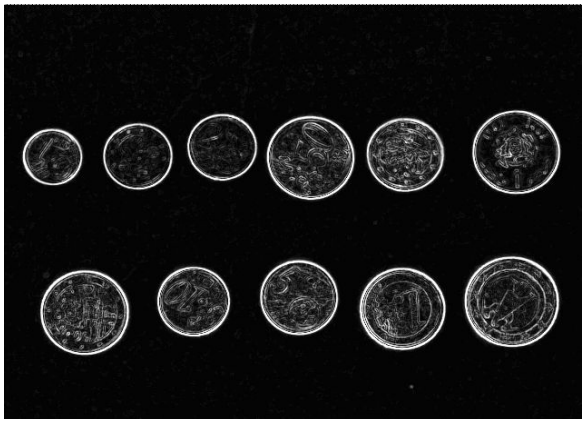


Figure 4: Result of the edge detection convolution operation

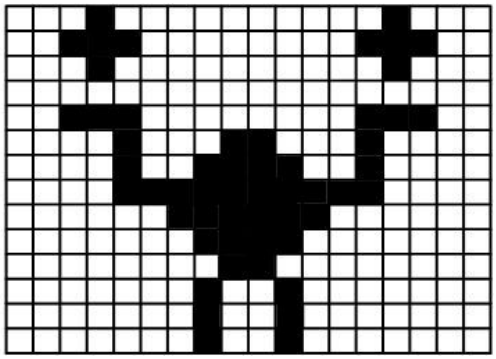


Figure 5: Example setup for finding connected regions

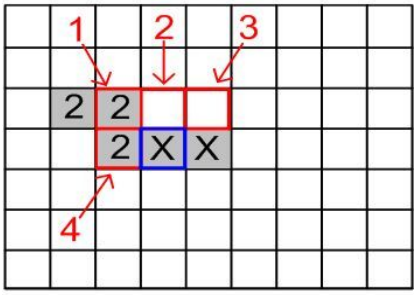


Figure 6: Operator for examining neighbouring regions (red border)

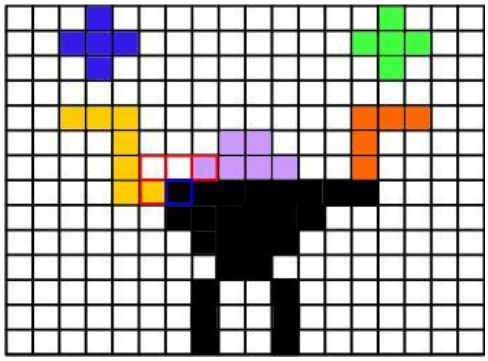


Figure 7: Conflicting region labels

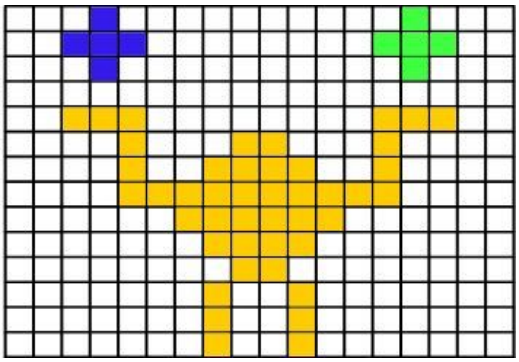


Figure 8: Result of the Connected Components Labelling algorithm

C. Connected Component Labelling

In order to identify regions that are connected, the Connected Component Labelling algorithm [3]-[4] is used. As can be seen in Figure 5 above, there are regions, that are directly connected and those, that are separated, as well as some regions, which are connected diagonally.

Most algorithms for finding connected components are developed for a software implementation and rely on the elegant formulation involving recursion, such as the flood-fill algorithm. For an FPGA implementation an iterative algorithm is needed.

The Connected Component Labelling is a direct iterative algorithm to find the regions connected together, which has a much longer formulation compared to the compact recursive flood-fill algorithm.

The neighbouring pixels are examined to check for already given labels. The 8-connectivity is used here to also accommodate for the possibility of diagonally connected region.

Only pixels, that have been found to be not part of the background during the segmentation, are checked. Since the operation is started from one of the corner points of the image, the labels are examined only for neighbouring pixels, which come before the current pixel. If a label is found, the current pixel being examined is given the same label. Otherwise, the label counter is incremented and the actual pixel is given the corresponding new label. At a certain point in the execution of the algorithm, it could come to the point, where two connected regions bearing different given labels are detected (Figure 7). In this case the smaller label is given to the pixel.

Having found that two regions with different given labels are connected, the algorithm now enters a re-labelling mode and re-labels the conflicting regions with the smaller label.

At the end of the of the connected region labelling it is found that the number of actual region is less than was suspected half way through the algorithm.

Although the Connected Component Labelling technique requires a more elaborate formulation than the flood-fill algorithm and substantially more difficult test and verification, for an FPGA hardware implementation, this iterative algorithm is needed.

D. Bounding Box Algorithm

For all connected regions, a best fit rectangle, known as the Bounding Box [3], can be drawn (Figure 9). This Bounding Box will help speed up search process later on.

Figure 9: Bounding Box drawn around the regions

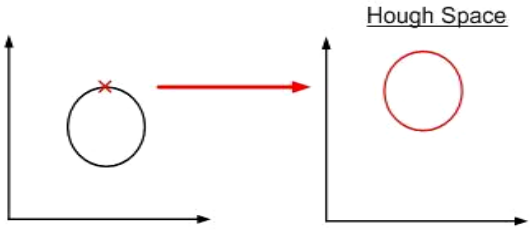


Figure 10: Transformation of a single pixel into a circle in the Hough Space

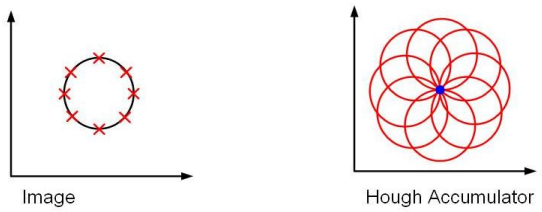


Figure 11: Hough Space after complete transformation of the single circle in the original space. Each circle in Hough Space belongs to another pixel.

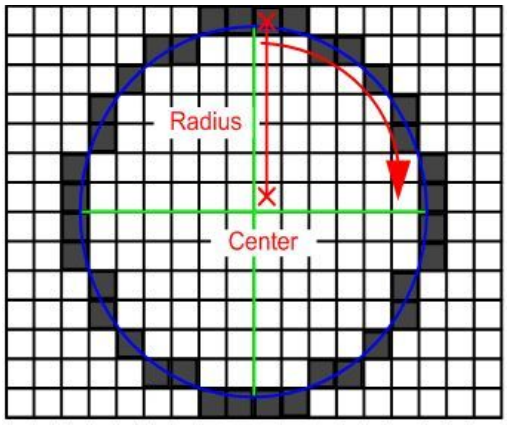


Figure 12: Bresenham's Midpoint Circle Algorithm

E. Circular Hough Transform

The Circular Hough Transform [5] is a technique used to find the centre of a circle, if the radius is known. Each pixel, which is not belonging to the background, is transformed into a circle in the Hough Space (Figure 10).

Pixel elements in the Hough Space (often alternatively denoted as "array elements") are incremented in their intensity by one step, if they belong to the Hough Space circle. This is done by checking each position in the array to see if the condition $R^2 = X^2 + Y^2$ is met.

When all the pixels of the image have been visited, the Hough Space is then searched for local maximum points. For a certain radius, these local maximum points indicate a possible circle centre (Figure 11).

This implementation with the checking of the condition $R^2 = X^2 + Y^2$ for each array element in Houch Space would need a lot of computation to be done. Keeping in mind, that the goal is an FPGA implementation, such a brute force solution would make things difficult and require a good deal of the FPGA resources.

Therefore, an alternative solution to the brute force version is needed for the Hough Transform and so one could utilize the Bresenham's Midpoint Circle Algorithm [6] to increment the points in the Hough Space in an efficient manner with much less computation (Figure 12).

As a benefit of this Midpoint Circle algorithm, not each point in the Hough Space array needs to be visited. Thus the required calculation is quite minimal. Defining a circle centre position $[X_C, Y_C]$ around which the circle has to be drawn, the algorithm is started at a position $[X_C, Y_C+R]$ and a decision is made, whether to move to the east or southeast. The decision is based upon a minimum error condition.

Each time the decision is made to move east, the X value is incremented and for the decision of moving to the southeast, the value of X is incremented and the value of Y is decremented.

This is done for one fourth of a circle i.e. one quadrant. By utilizing symmetry, the points for the other three quadrants are plotted, without the demand to invoke the decision process above again. Thus, further speed-up of the algorithm is achieved.

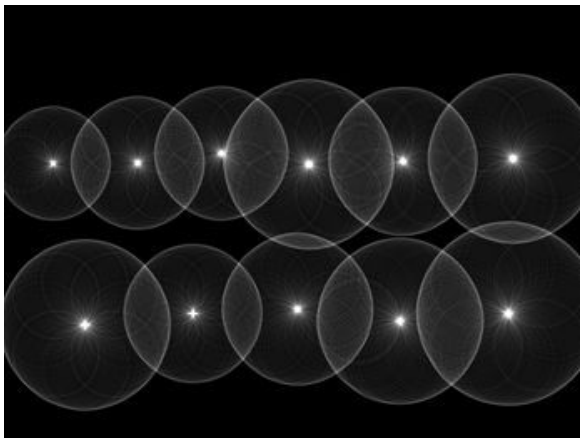


Figure 13: Result of the test setup from Figure 1 in the Hough Space

Figure 14: Circles and midpoints indicate the results, which are superimposed to the original test setup picture

The Circular Hough Transform is applied to the various regions within the respective bounding boxes, when searching for circles of specific radii. Seen above is the corresponding result of the VHDL implementation in the Hough Space (Figure 13).

After the Circular Hough Transformation, the search for the local maxima is done to determine the respective circle centre. Result from the bounding box could be used accelerate location of local maxima. In Figure 14 the results (circle and midpoint) of the original test setup from Figure 1 are given.

III. Development Flow of Hardware Blocks

As seen before, in order to develop an Image Processing and Pattern Recognition application, it is first necessary to determine the needed algorithm for each purpose and lay out the sequence of processing of the data.

Therefore careful thought has to be put into the algorithms to be implemented, both pre-processing and target algorithm, because FPGA resource consumption

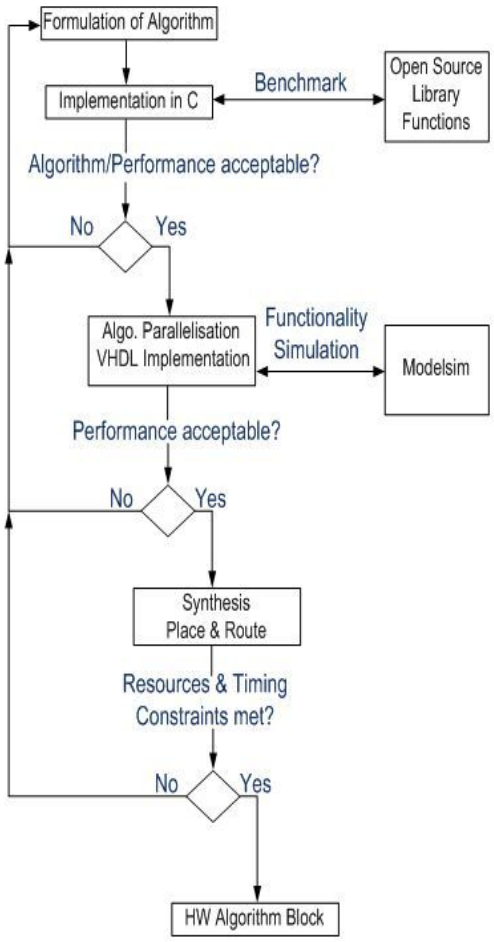


Figure 15: HW-Algorithm Development Flow Chart

and development time are dependent on the decisions in the early development phase. Each algorithm has to be developed into a hardware block, which as a unit fulfils the defined task.

A suitable pathway for development of the algorithm is to first formulate it iteratively, then implement it in a software programming language such as C. This is done to test the functionality and performance of the algorithm.

For this purpose also Image Processing library modules, such as that from OpenCV, can be used as a benchmarking platform to test. The result from the self written algorithm can be compared and validated in terms of quality and execution speed.

After this stage is cleared, and the performance and execution time is acceptable, then the code is formulated in a Hardware Description Language such as VHDL. Here the parallel computing abilities of the FPGA should be utilized. For example, some of the operations can be carried out in parallel, such as the convolution operation in the Sobel edge detection, where all the multiplications of the edge detection mask can be performed in one clock cycle (Figure 16).

```
for i in -1 to 1 loop
  for j in -1 to 1 loop  -- Convolution
    new_x := new_x + sobel_x(i,j)*((pre_sobel_data(i,j)));
    new_y := new_y + sobel_y(i,j)*((pre_sobel_data(i,j)));
  end loop;
    if i = 1 then
      --  Gradient detection in X & Y Direction
      value_x_y := (abs(new_x)+abs(new_y))/8 ;
      if value_x_y > 255 then
        value_x_y := 255;
      end if;
      post_sobel_data  := value_x_y;
      new_x := 0;
      new_y := 0;
      coloumn := coloumn + 1;
    end if;
end loop;
```

Figure 16: VHDL Code for parallelization of Sobel edge detection algorithm

Table 3: Bounding Box results with C

Bounding Box (C Implementation)								
	Top left		Top Right		Bottom Left		Bottom Right	
	Y	*X*	*Y*	*X*	*Y*	*X*	*Y*	*X*
1	122	518	122	612	217	518	217	612
2	126	204	126	278	201	204	201	278

Table 4: Bounding Box results with VHDL

Bounding Box (VHDL Implementation)								
	Top left		Top Right		Bottom Left		Bottom Right	
	Y	*X*	*Y*	*X*	*Y*	*X*	*Y*	*X*
1	122	518	122	612	217	518	217	612
2	126	204	126	278	201	204	201	278

The same is possible for the operations of the segmentation process.

IV. PERFORMANCE CROSSCHECK BETWEEN C AND VHDL

For the pre-processing algorithms, an example comparison between the C routine and VHDL is presented below. In this case the results are identical (Table 3 and Table 4).

Table 5: Hough Transform Results C

Circle Center (C Implementation)		
	X	*Y*
1	565	170
2	241	164
3	338	176
4	443	173

Table 6: Hough Transform Results VHDL

Circle Center (VHDL Implementation)		
	X	*Y*
1	565	170
2	241	164
3	339	176
4	442	173

Next for the target algorithm, in this case the Hough Transform, an example comparison is given in Table 5 and Table 6.

As can be seen, the results of the algorithm implemented in C and VHDL are very concurrent. Only in a few scenarios the values differ by one pixel, e.g. the centre of the circle found after the Hough Transform in region 3 and 4. This, however, is quite minimal and does not have any severe effects on the end results in our case.

V. DATA STORAGE & TRANSFER BETWEEN ALGORITHMS

Image Processing not only has a large amount of data to be processed, but also a large amount of data, that has to be stored and transferred between the algorithms. Here the question arises about the size of the required on-chip memory of the FPGA.

If the FPGA has sufficient on-chip memory, then the design can be implemented more efficiently as compared to the case where an external memory is needed. The latter case will of course have a detrimental effect on the processing speed.

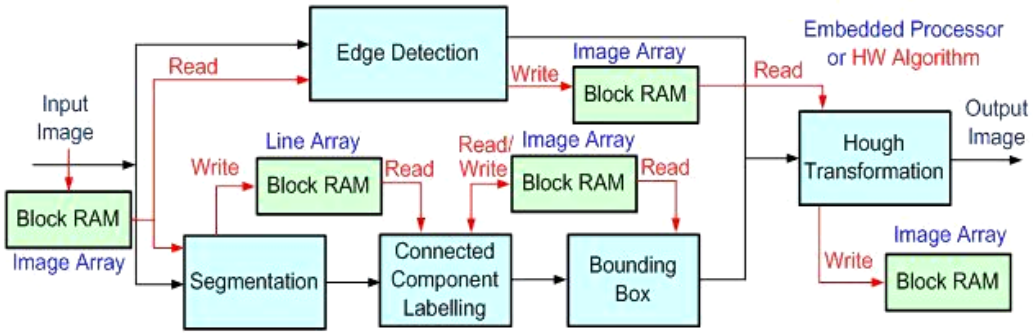


Figure 17: Design with on-chip RAM (Block RAM of an FPGA)

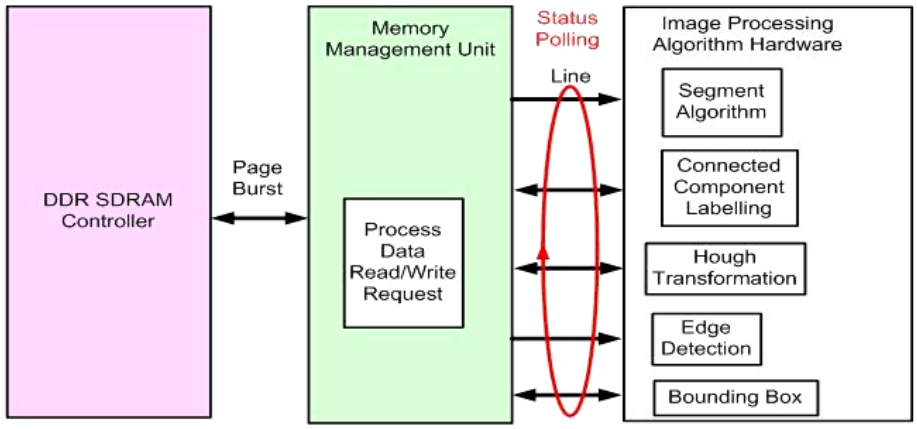


Figure 18: Design with off-chip RAM (in this case DDR SDRAM) requiring a Memory Management Interface

A. Internal Data Storage

If the FPGA has sufficient Block RAMs available, then memory storage can be forseen for each of this algorithm blocks. This also makes it easier to transfer data between the algorithm blocks. Depending on the algorithm, we do not always have to allocate memory the size of the image itself.

With some synchronization of operation between two blocks, it would suffice to have a line array only (e.g. the line array in Figure 17 for Segmentation (Read) and Connected Component Labelling (Write)). This would lessen the demand of memory allocation, increasing efficiency. Due to the nature of the Hough Transform algorithm, which is difficult to be computed parallely, an embedded microprocessor can also be used to perform the calculations.

B. External Data Storage

As can be seen in Figure 18, each single algorithm block has to read and write to an external data storage unit, if the FPGA does not have sufficient on-chip memory. For handling the read/write requests of each of these algorithms, a Memory Management unit is implemented.

In this type of implementation, each algorithm has to await its turn to read or write data and the Memory Management Unit sequentially goes through each pending request. This creates bottlenecks for the ability of the algorithms to process data independently from one another.

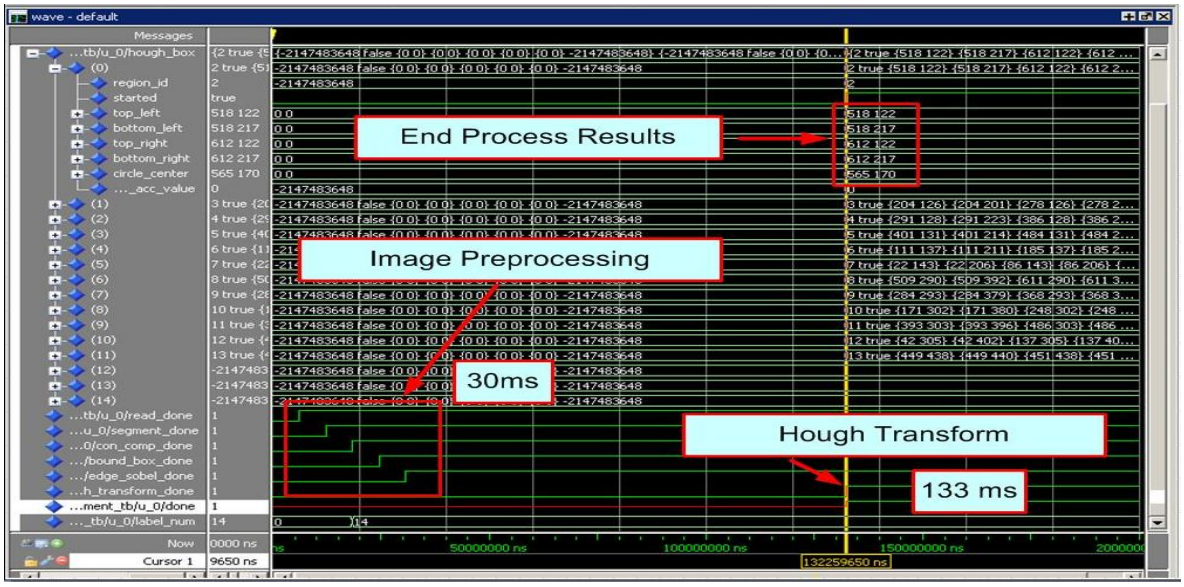


Figure 19 Modelsim simulation result

```
----------------------
Family : Virtex2p
Device : xc2vp30
Package : ff896
Speed : -7
----------------------

Timing Summary:
---------------
Speed Grade: -7

  Minimum period: 9.498ns (Maximum Frequency: 105.282MHz)
  Minimum input arrival time before clock: 11.129ns
  Maximum output required time after clock: 3.461ns
  Maximum combinational path delay: No path found
```

```
=========================================
Advanced HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                : 116
 10-bit adder                      : 5
 10-bit subtractor                 : 10
 32-bit adder                      : 41
 5-bit adder                       : 5
 5-bit addsub                      : 5
 5-bit subtractor                  : 45
 6-bit subtractor                  : 5
# Counters                          : 22
 32-bit up counter                 : 22
# Registers                         : 2370
 Flip-Flops                        : 2370
# Comparators                       : 105
 32-bit comparator greatequal      : 53
 32-bit comparator less            : 12
 5-bit comparator greatequal       : 5
 5-bit comparator greater          : 25
 5-bit comparator lessequal        : 5
 8-bit comparator greater          : 5

=========================================
```

Figure 20: Synthesis results with Precision synthesis tool for the pre-processing algorithms.

VI. Simulation & Synthesis Results

Simulation results for the on-chip memory design are shown in Figure 19 and the synthesis results in Figure 20. It can be seen from the Modelsim simulation, that the image pre-processing algorithm requires less computation time as compared to the circular Hough Transform Pattern Recognition algorithm. This is due to the fact that the Hough Transform not only is computationally intensive, but also is difficult to compute in parallel. Hence it is clear, that longer computation time is needed.

The synthesis results are given for the pre-processing algorithms only. Here the maximum clocking frequency and also the amount of hardware required to implement these pre-processing hardware algorithms can be retrieved. Due to limited resources of the platform available for this work, on-chip memory was excluded from the synthesis.

VII. Associated Algorithm Development Challenges

There is a certain degree of challenge when developing the Image Processing and Pattern Recognition algorithms for a hardware implementation compared to the software development. First of all, a solid understanding of the algorithm is required in detail.

This also is necessary when later formulating a suitable version of the algorithm for hardware implementation, especially when making use of the FPGA parallel computing capability. It is also important to avoid large multiplications and the division operation (if

```
if segment_data(line,coloumn) = 255 and region_data(line,coloumn) = Label_i then
  --initialize
  if radius_calc_init = false then
    radius := (box_i(Label_i-2).top_right.x_pos - box_i(Label_i-2).top_left.x_pos)
    x :=0;
    y := radius;
    d := 1 - radius;
    radius_calc_init := true;
    Max_Acc:=0;
  end if;
  if x<y then
    -- Increment Points Routine here,for center x+xc & y+yc
    if d<0 then
      x := x+1;
      d := d+2*x+3;
    else
      x := x+1;
      y := y-1;
      d := d+2*(x-y)+5;
    end if;
  end if;
  if x>=y then
    coloumn := coloumn +1;
    x :=0;
    y := radius;
    d := 1 - radius;
  end if;
else
  coloumn := coloumn +1;
  x :=0;
  y := radius;
  d := 1 - radius;
end if;
```

Figure 21: Hough Transform using Bresenham algorithm routine abstract

possible), since this slows things down and increases hardware resources consumption.

For e.g as we have seen in an example, it is advantageous to utilize the Bresenham's Midpoint Circle algorithm to draw the circle in the Hough Transform instead of checking, if condition $R^2 = X^2 + Y^2$ is met via brute force.

Moreover, the algorithm has to be carefully designed in order to be executed within each clock cycle. If the clock cycle is not optimally utilized, then the operations would require more cycles, slowing down the whole process.

VIII. Conclusion

In this work, the principle, development, implementation and testing of Image Processing and Pattern Recognition in a FPGA has been examined. Despite the parallel computing capabilities of an FPGA, there are still downsides in terms of memory for storage of the large amount of data. However with the rapid improvement in the FPGA technology, this disadvantage is quickly disappearing and also mid-cost FPGA have the required on-chip memory, e.g. Kintex-7 [7]. Therefore, in the coming years embedded Image Processing in FPGAs will play an even more important role.

Acknowledgement

We would like to thank Prof. Dr. U. Klauck and Dipl.-Ing. J. Hahn-Dambacher for their time and constructive discussion sessions, which shed new light on various key matters. References [8]-[9] provided a base for discussions on implementation issues.

References

[1] Bradski, G. et. al.: Learning OpenCV : Computer Vision with the OpenCV Library, O'Reilly 2008.

[2] Laganière, R.: OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing 2001.

[3] Klauck, U.: Bildverarbeitung, Manuskript, Hochschule Aalen 2010

[4] Jankowski, M., Kuska J.-P.,: Connected components labeling-algorithms in Mathematica, Java, C++ and C#. http://www.izbi.uni-leipzig.de/izbi/publi_2004/IMS2004 /IMS2004_JankowskiKuska.pdf . Date of access: July 2011

[5] Borovicka, J.: Circle Detection Using Hough Transforms 2003. http://linux.fjfi.cvut.cz/~pinus/bristol/imageproc /hw1/report.pdf. Date of access: July 2011

[6] Kalra, P.K.: Graphics System-Raster Graphics, IIT Delhi. http://www.cse.iitd.ernet.in/~pkalra/cs474/slides/lecture2+3 .PDF: Date of access: July 2011

[7] http://www.xilinx.com/products/silicon-devices/fpga/kintex-7/index.htm, Date of access: July 2011

[8] Klupsch, S. et. al.: Real Time Image Processing based on Reconfigurable Hardware Acceleration, http://www.mpiinf.mpg.de/~strzodka/papers/public/KlErHu_ 02fpga.pdf, Date of access: July 2011

[9] Nelson, A.: Implementation of Image Processing Algorithms on FPGA Hardware, Master Thesis, Vanderbilt University, May 2000, citeseerx.ist.psu.edu, Date of access: July 2011

Simran Singh received his academic degree Dipl.-Ing.(FH) in electronics in December 2009 and is currently pursuing his Master's Degree in Computer Controlled Systems both at the University of Applied Sciences Aalen. He is currently finalizing his Thesis in Digital Correction Algorithms for Analog Nonlinearities.

Heinz-Peter Bürkle received the academic degree Dipl.-Ing. in electrical engineering in 1991 from the University of Stuttgart and the degree of Dr.-Ing. in 1997 also from the University of Stuttgart. After several years in R&D at Alcatel he is a Professor of semiconductor technology, circuit design and computer-aided circuit design at the Aalen University of Applied Sciences since 2003.

HOCHSCHULE KONSTANZ
TECHNIK, WIRTSCHAFT UND GESTALTUNG

Applikationsspezifischer Softcore-Prozessor für sicherheitskritische Embedded-Systeme

Kai Sascha Brach, Hannes Hennig, Christian Kielmann, Kamil Wistuba, Irenäus Schoppa

***Zusammenfassung*—Die Synthese einer neuen Prozessorarchitektur ist eine anspruchsvolle und herausfordernde Aufgabe, besonders dann, wenn es darum geht, eine neuartige Architektur zu entwerfen, zu modellieren und auf einem FPGA zu implementieren. Im Rahmen eines Teamprojektes des Informatik-Master-Studienganges an der HTWG Konstanz wurde ein 16-Bit-Softcore-Prozessor namens Icarus entwickelt. Dieser Prozessor wurde als Speicher-Speicher-Architektur speziell für sicherheitskritische Aufgaben im Bereich der eingebetteten Systeme konzipiert. Die Implementierung erfolgte auf der Register-Transfer-Ebene in VHDL, und wurde erfolgreich auf einem Entwicklungsboard mit einem Spartan 3 Baustein getestet. Zusätzlich wurde ein Assembler in C++ mitentwickelt.**

***Schlüsselwörter*—Softcore-Prozessor, Eingebettete Systeme, CISC, FPGA, Mikroprogrammierte Kontrolleinheit, Multitasking**

I. Einleitung

Moderne Prozessoren nutzen die bewährte Harvard-Architektur, welche zwischen Programm- und Datenspeicher unterscheidet. Dieser Unterschied resultiert in einem separaten Programm- und Datenbus. Der Vorteil dieser Architektur ist, dass im Speicher, im Gegensatz zu der Von-Neumann-Architektur, nicht zwischen Programm- und Datensegmenten unterschieden werden muss, da diese Segmente in unterschiedlichen physikalischen Bereichen liegen. Die klassischen Architekturklassen verarbeiten Operanden in Speichern und Registern (Register-Speicher-Architekturen) oder in Registern (Register-Register-Architekturen). Der Vorteil der Register-Register-Architekturen, auch Load/Store-Architektur genannt, ist der besonders schnelle Zugriff auf Daten, die sich in Registern befinden, sowie die Möglichkeit, die Verarbeitung mit einem Fließband zu realisieren. Der Nachteil dieser Architektur besteht in der begrenzten Anzahl an Registern, z. B. 16 bis 256.

K. Brach, kabrach@htwg-konstanz.de, H. Hennig, hahennig@htwg-kostanz.de, Ch. Kielmann, chkielma@htwg-konstanz.de und K. Wistuba, kawistub@htwg-konstanz.de, sind Studenten an der HTWG Konstanz, I. Schoppa, ischoppa@htwg-konstanz.de, ist Mitglied der HTWG Konstanz, Brauneggerstr. 55, 78462 Konstanz.

Die heutigen FPGAs enthalten zahlreiche RAM-Blöcke, deren Zugriffszeiten vergleichbar mit den Zugriffszeiten auf Flipflops und Register sind. Beispielsweise ermöglichen die Spartan 3-Bausteine Zugriffszeiten von 5 ns auf die RAM-Blöcke [1, 2].

Der hier vorgestellte experimentelle Prozessor basiert auf einer Speicher-Speicher-Architektur, die durch die schnellen Zugriffszeiten auf die RAM-Blöcke eine hohe Effizienz ermöglicht. Grundlage des Prozessors sind in Hardware implementierte Tasks, die über einen Dispatcher verwaltet werden und jeweils eine eigene Interrupt-Quelle besitzen. Zudem unterstützt die Architektur sowohl eine Kommunikation zwischen einzelnen Tasks, als auch eine Echtzeit-Taskumschaltung auf Hardware-Ebene. Der Speicherschutz des Prozessors wird durch die Segmentierung der einzelnen Speicherblöcke gewährleistet. Somit ist es nicht möglich, dass eine Task in den Speicherbereich einer anderen Task schreibt.

Des Weiteren ist der Icarus Softcore-Prozessor speziell für den Bereich der eingebetteten Systeme konzipiert, indem er das Prinzip des Multitaskings mit bis zu 32 Tasks unterstützt. Die Anzahl der Tasks kann zur Laufzeit nicht verändert werden, wodurch die dynamische Allokation von Speicher entfällt. Die Echtzeit-Taskumschaltung wird mit einem Interrupt-Controller prioritätengesteuert realisiert, indem die taskspezifischen Informationen gesichert und geladen werden. Für die Intertaskkommunikation wurde ein einheitliches Konzept implementiert, welches aus einer taskeigenen Nachrichtenwarteschlange, sowie einem Message-Protokoll besteht.

Für einen Überblick über die grundlegenden Prinzipien von Mikrocontrollern bzw. Mikroprozessoren eignet sich [3], welches den aktuellen Stand der Mikroprozessortechnik, sowie zukunftsweisende Entwicklungstendenzen und Forschungen vorstellt. Weiterhin sind in [4] die hardware-technischen Merkmale und Implementierungen von modernen Rechnerarchitekturen aufgezeigt. Diese werden mit modernen leistungserhöhenden Maßnahmen anhand von relativ neuen Prozessoren vertieft. [5] behandelt einfache bis hin zu komplexen Spezialprozessoren und deren zugehörige Modulbausteine, wie Speicher und Interrupt-Controller. Zudem werden alle Entwicklungsschritte von der Planung bis zum Betrieb eines Mikroprozessors beschrieben.

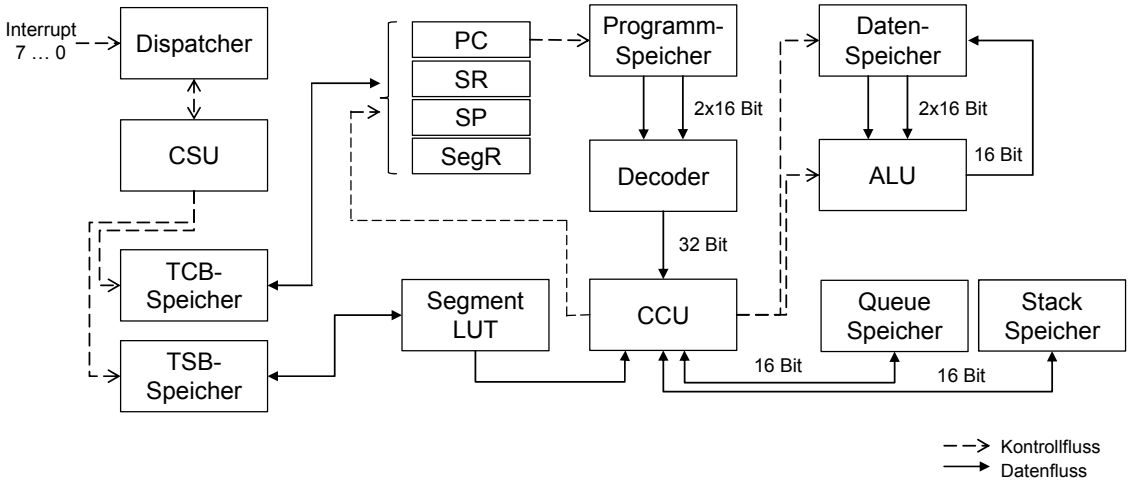


Abbildung 1: Schematische Architektur des Softcore-Prozessors.

II. Architektur

Die Architektur des Softcore-Prozessors ist in Abbildung 1 dargestellt und besteht aus einer Harvard-Architektur mit getrenntem Daten- und Programmspeicher. Beide Speicher sind als 16 Bit Dual-Port-Speicher realisiert. Auf den Programmspeicher wird nur lesend und auf den Datenspeicher lesend, sowie schreibend zugegriffen. Des Weiteren gibt es separate Speicher für die Taskverwaltung Task Control Block (TCB) und Task Segment Block (TSB), sowie für die Queue und den Stack. Der Prozessor verfügt zudem über vier Steuerregister: den Programm Counter (PC), das Status Register (SR), den Stack Pointer (SP) und das Segment Register (SegR). Der PC, das SR und der SP fungieren analog zu herkömmlichen Rechnerarchitekturen. Das SegR dient zur Verwaltung der Segmente innerhalb eines TSBs bzw. der Segment-Lookup-Table (Segment-LUT). Aufgrund der Dual-Port-Architektur des Programmspeichers erfolgt ein paralleler Zugriff auf die beiden 16 Bit Datenworte des 32 Bit Befehlsformats. Die funktionalen Einheiten, wie Dispatcher, Arithmetic Logical Unit (ALU), Context Switch Unit (CSU), Command Control Unit (CCU) und der Decoder werden in VHDL modular beschrieben und sind in den folgenden Unterkapiteln erklärt.

A. Arithmetic Logical Unit (ALU)

Die ALU verarbeitet die Rechen- und Logikoperationen des Prozessors und kann zwei 16 Bit Operanden für Quelle und Ziel gleichzeitig verarbeiten. Zu den Grundoperationen der ALU gehören die Addition, Subtraktion, sowie Vergleiche und logische Operationen. Das Ergebnis der ALU wird an die Adresse des Ziel-Operanden zurück geschrieben.

B. Command Control Unit (CCU)

Die CCU ist das Steuerwerk des Icarus und korrespondiert direkt mit der sequentiellen Abarbeitung der Instruktionen. Der Entwurf des Steuerwerks basiert auf der Grundlage eines Mikroprogrammspeichers. Dabei wird ein Steuerungsalgorithmus als Programm im Mikroprogrammspeicher abgelegt und als Folge von Mikrobefehlen abgearbeitet. Der Mikroprogrammspeicher besteht aus einem synchronen RAM-Block mit einer Breite von 72 Bit (inkl. 4 Parity-Bits) und 128 Adressen. Die besondere Breite des Mikroprogrammspeichers ist notwendig, um alle Eingangsbedingungen parallel verarbeiten zu können. Da der RAM-Block als synchroner Speicher ausgelegt ist, wurde diese Eigenschaft beim Entwurf des Mikroprogramms berücksichtigt. Dazu erfolgt die Realisierung des Mikroprogramms als Mealy-Automat, unter Berücksichtigung, dass der Steuerbefehl nach einem Takt an den Ausgangsflipflops zur Verfügung steht.

C. Decoder

Der Decoder bekommt den Befehl aus dem Programmspeicher übergeben und verarbeitet die im Befehlswort stehenden Informationen. Das Befehlswort ist 32 Bit lang und wird aus dem Programmspeicher in den Decoder eingelesen. Anschließend erfolgt die Selektierung des Opcodes, welcher als Einsprungsadresse in das entsprechende Mikroprogramm dient. Im Mikroprogramm sind die Anzahl der Operanden, die Adressierungsart sowie Steuersignale zur weiteren Abarbeitung des Befehls definiert.

D. Dispatcher

Der Dispatcher ist ein prioritätengesteuerter 7-stufiger Interrupt-Controller. Sämtlichen Tasks wird bei der Initialisierung eine Priorität zugewiesen. Jeder Prioritätsstufe definiert einen eigenen Hardware-Interrupt, wie bspw. UART- oder Timer-Interrupts.

Wie in Abbildung 3 dargestellt, enthält der Dispatcher zu jeder Prioritätsstufe eine Liste, die die zugeordneten Tasks beinhaltet. Sind einer Prioritätsstufe mehrere Tasks zugeordnet, erfolgt bei jedem Interrupt ein zyklischer Wechsel zwischen den Tasks. Der Dis-

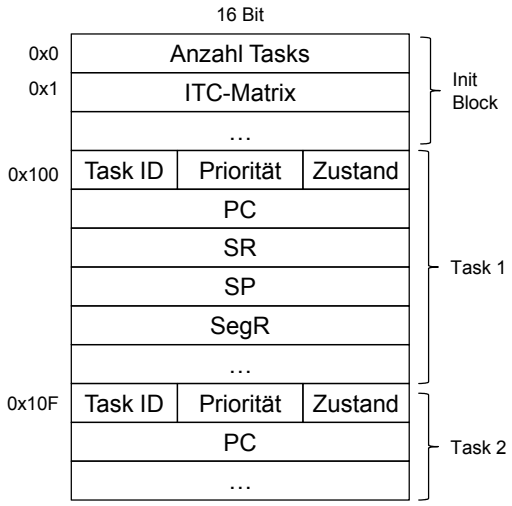


Abbildung 2: Aufbau des Task Control Blocks (TCB).

patcher fungiert zudem als Scheduler, da er die lauffähigen Tasks von abgeschlossenen oder unterbrochenen Tasks unterscheidet. Die Task-Listen des Dispatchers sind als LUTs aufgebaut. Neben den Task-Listen wird ein Interrupt-Stack mitgeführt, indem sämtliche unterbrochenen Tasks gespeichert werden. Tasks können entweder durch Interrupts oder durch Unterprogrammaufrufe unterbrochen werden. Der Interrupt-Stack erlaubt die Abarbeitung eines kompletten Unterbrechungspfades in umgekehrter Reihenfolge.

Der Kontextwechsel erfolgt mit Hilfe der CSU, die das Laden und Sichern der physikalischen Steuerregister aus bzw. in die Speicher vornimmt. Während eines Kontextwechsels fungiert der Dispatcher als zusätzliches Steuerwerk und unterbricht die Verarbeitung der CCU, damit die zu sichernden Steuerregister in einem definierten Zustand bleiben. Im Anschluss an einen Kontextwechsel wird die CCU vom Dispatcher wieder aktiviert.

E. Context Switch Unit (CSU)

Die CSU führt den Wechsel des Taskkontextes physikalisch durch und wird vom Dispatcher gesteuert. Hierzu werden die internen Steuerregister der unterbrochenen Tasks in dem entsprechenden Bereich der TCB gesichert. Anschließend werden die Steuerregister, sowie die Segment-LUT der darauffolgenden Task geladen.

F. Task Segment Block (TSB)

Der TSB enthält die Deskriptoren der Tasks, die mit der Segmentadresse konkateniert werden und somit die effektive Adresse bilden. Des Weiteren erfolgt auch eine Einteilung der benötigten Segmente für den Programm-, Daten, Stack- und Queuespeicher. Für jede Task können maximal 16 Segmente vergeben werden. Die Deskriptoren werden fortlaufend im TSB abgelegt, die Zugehörigkeit zur Segmentgruppe wird

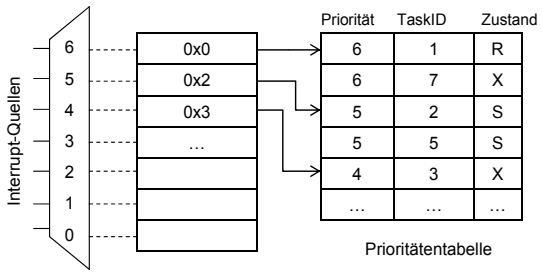


Abbildung 3: Schematischer Aufbau des Dispatchers.

in dem SegR gespeichert. Der schematische Aufbau des TSB ist in Abbildung 4 dargestellt.

G. Task Control Block (TCB)

Der TCB wird für die Sicherung der Taskzustände bei einem Kontextwechsel, sowie deren Wiederherstellung benötigt. Er besteht aus einem globalen Block, der die Anzahl der Tasks und die Intertaskcommunication (ITC)-Matrix beinhaltet, wie Abbildung 2 zeigt. Des Weiteren folgen die Kontrollblöcke der einzelnen Tasks. Jeder Kontrollblock beinhaltet die Task ID, die Priorität und den Taskzustand. Zudem wird für jede Task der PC, das SR, der SP und das SegR gesichert.

H. Segment-LUT

Die Segment-LUT beinhaltet die Deskriptoren der einzelnen Segmente aus dem TSB der aktuellen Task. Mit Hilfe des SegR werden die Segmente den Seg

mentgruppen zugeordnet. Der Decoder konkateniert den Deskriptor des aktuellen Segments aus der Segment-LUT mit der Adresse des PCs oder des Adressoperanden im Befehlswort. Die effektive Adresse, sowie der effektive PC setzen sich somit aus der Konkatenation des 8 Bit Deskriptors und der 8 Bit Segmentadresse zusammen. Dadurch wird sichergestellt, dass jede Task nur Zugriff auf ihren zugeordneten Speicherbereich hat. Aufgrund der Konkatenation ist die Adressierung von 2^{16} Bytes im Speicher möglich. Da die Deskriptoren in der Segment-LUT abgelegt sind, erfolgt dies innerhalb eines Taktes.

III. INTERTASKCOMMUNICATION (ITC)

Die ITC bezeichnet die Kommunikation und den Datenaustausch zwischen zwei oder mehreren Tasks. Für die Task-Kommunikation wurde ein einheitliches Kommunikationskonzept entworfen, welches auf den Grundprinzipien des Message Passing beruht. Um den Datenaustausch zwischen den Tasks zu gewährleisten stehen mehrere Verarbeitungsmechanismen, ein Kontrollmechanismus in Form der ITC-Matrix und ein fest definiertes Message-Protokoll zur Verfügung.

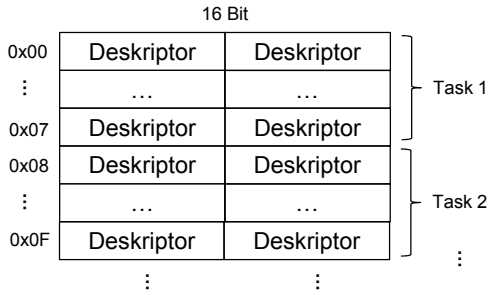


Abbildung 4: Aufbau des Task Segment Blocks (TSB).

A. Kommunikationsablauf

Die Kommunikation zwischen den Tasks beruht auf der Übertragung von Daten oder Referenzen. Bei der reinen Datenkommunikation werden reine Daten von einer Task in die Queue einer anderen Task gesendet. Diese Daten werden bei der Verarbeitung der Empfänger-Task in den entsprechenden Bereich des Datenspeichers kopiert. Diese Art der Kommunikation wird bevorzugt bei kleinen Datenmengen oder bei der Kommunikation zwischen hoch priorisierten Tasks verwendet.

Bei der Kommunikation mittels Referenzen wird der Empfänger-Task mitgeteilt, dass ein freigegebener Bereich im Datenspeicher der Sender-Task zur Datenabholung für sie reserviert ist. Sobald die Task dem Prozessor zugeteilt wurde, kann sie die für sie reservierten Daten abholen und in ihren eigenen Datenspeicher kopieren. Für das Abholen der Daten steht ein Datenkontroller zur Verfügung. Nach dem vollständigen kopieren der Daten teilt die Empfänger-Task der Sender-Task mit, dass der Datenbereich wieder freigegeben werden kann. Die Kommunikation mittels Referenzen wird bevorzugt bei großen Datenmengen oder zur Kommunikation mit niedrig priorisierten Tasks verwendet.

Für die Initiierung zur Verarbeitung der Queue-Daten steht ein atomarer Look in Queue (LIQ) Befehl zur Verfügung. Der LIQ Befehl ruft den Datenkontroller zur Interpretation des in der Queue befindlichen Message-Protokolls auf (siehe Abschnitt B).

B. Message-Protokoll

Für das Senden von Daten oder Referenzen werden separate Message-Protokolle verwendet. Das Message-Protokoll für das Quittieren von bereits abgeholten Nachrichten entspricht demselben Protokoll zum Senden von Referenzen. Das Senden und Quittieren von Nachrichten wird als Handshaking bezeichnet. Abbildung 5 zeigt das Message-Protokoll für das Senden von Referenzen und Daten.

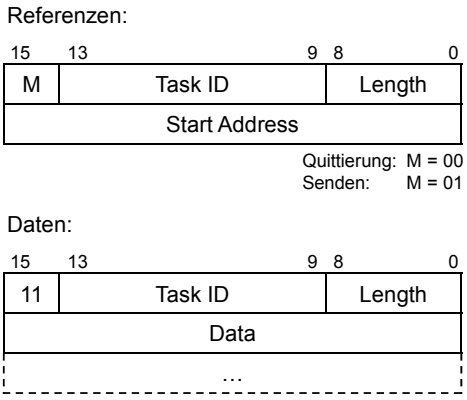


Abbildung 5: Message-Protokoll für die Kommunikation mit Daten und Referenzen.

C. Spezialbefehle

Für die Kommunikation und Synchronisation zwischen Echtzeit-Tasks stehen in herkömmlichen Prozessoren üblicherweise Mutexe, Semaphore und Mailboxen zur Verfügung [6, 7]. Damit auf diese Mechanismen verzichtet werden kann, wurden diese Prozeduren als atomare Befehle SENDR/SENDD und LIQ entwickelt, welche nicht unterbrochen werden können. Die beiden unterschiedlichen Kommunikationsarten die durch Daten und Referenzen gegeben sind, wurden durch die beiden Befehle SEND Reference (SENDR) und SEND Data (SENDD) abgebildet. Beide Befehle benötigen die Task ID des Empfängers, sowie die zu sendenden Daten und deren Länge oder als Referenz die Startadresse und die Länge der Daten im taskeigenen Datenspeicher, wie das Message-Protokoll in Abbildung 5 zeigt. Eine vollständige Datenübertragung ist gewährleistet, indem die Befehle nicht unterbrechbar sind.

D. Queue Verwaltung

Innerhalb des Queue-Speichers einer Task sind die ersten 16 Adressen zur Verwaltung der Kommunikation zwischen den Tasks reserviert. Werden Referenzen gesendet, wird eine Kopie des Message-Protokolls innerhalb dieses Bereiches zur Verwaltung abgelegt. Die Kopie ist für die Blockierung und die Freigabe des Datenspeichers erforderlich.

E. ITC-Matrix

Die ITC-Matrix dient zur Überprüfung der Kommunikationsrichtung. Sie wird vom Programmierer definiert und kann zur Laufzeit nicht verändert werden. Mit der ITC-Matrix kann eine Sender-Task prüfen ob sie Nachrichten an eine bestimmte Task senden darf.

Empfänger →
Sender ↓

Tasks#	1	2	3	…	32
1	1	0	0	…	1
2	0	1	1	…	0
3	1	0	0	…	0
…	…	…	…	…	0
32	0	0	0	0	0

Abbildung 6: ITC-Matrix für 32 Tasks.

Die Empfänger-Task hat ihrerseits die Möglichkeit die Kommunikationsrichtung zu prüfen. Abbildung 6 zeigt den schematischen Aufbau der ITC-Matrix. Die Kommunikationsrichtung ist über die Zeilen und Spalten der Matrix definiert. Die Spalten bilden die erlaubten Empfänger-Tasks und die Zeilen die Sender-Tasks ab. Für Task 2 ist bspw. definiert, dass sie an Task 3 senden und von keiner Task empfangen darf. Der Datencontroller verwendet die ITC-Matrix zur Überwachung der Kommunikation. Im Fehlerfall, wenn einer unerlaubten Task Daten gesendet werden sollen, wird ein Fehlerflag im Statusregister des Prozessors gesetzt.

IV. Befehle

Im folgenden Kapitel werden der Befehlssatz, das Befehlsformat und die Adressierungsarten aufgeführt. Der Befehlssatz umfasst 28 Befehle, sowie drei Adressierungsarten im 2-Adressformat. Das Befehlsformat besitzt eine Form die auf die Segmentierung des Speichers mit Deskriptoren angepasst wurde.

A. Befehlssatz

Neben sieben arithmetischen und vier logischen Befehlen sind vier Bitmanipulations-, sowie ein Datentransport-, fünf Sprung-, drei Kommunikations- und vier sonstige Befehle implementiert. Alle weiteren Befehle zur Kommunikation und Initialisierung wurden als Assembler-Direktiven implementiert. Tabelle 1 zeigt eine Auflistung des Befehlssatzes.

B. Befehlsformat

Das Befehlsformat besteht aus 32 Bit, die sich einheitlich für alle Befehle aus dem 8 Bit Operationscode, gefolgt von einem Quelloperanden, bestehend aus 4 Bit Deskriptor für die Kodierung des Quellsegments und 8 Bit Quelladresse. Der im Anschluss folgende Zieloperand besitzt dasselbe Format mit einem 4 Bit Deskriptor für das Zielsegment und 8 Bit Zieladresse.

C. Adressierungsarten

Der Icarus Prozessor unterstützt die drei Adressierungsarten direkt, indirekt und unmittelbar. Bei der

Tabelle 1: Befehlssatz

Gruppe	Befehl
Arithmetik	ADD, SUB, ADC, SUBC, INC, DEC, CMP
Logik	NOT, AND, OR, XOR
Schiebebefehle	ROR, ROL, SHR, SHL
Transport	MOV
Sprungbefehle	JAL, BNC, BNN, BNZ, BNO
ITC	SENDR, SENDD, LIQ
Sonstige	BRK, NOP, IRE, IRD

direkten Adressierung werden die Speicherzellen direkt über die Adressoperanden des Befehlswortes adressiert. Bei der indirekten Adressierung verweist die Speicherzelle des Adressoperanden auf eine weitere Adresse im Datenspeicher. Die indirekte Adressierung wird insbesondere für die effektive Umsetzung von Zustandsautomaten verwendet. Hierbei entspricht jeder Zustand einer Speicherzelle die auf eine Funktion in Hauptspeicher referenziert. Bei der unmittelbaren Adressierung steht ein konstanter Wert in den Adressoperanden.

V. Messungen

Der Kontextwechsel einer Task wird bei einer Taktfrequenz von 50 MHz innerhalb von 560 ns durchgeführt. Die Auswahl einer neuen Task wird dabei vom Dispatcher innerhalb von 80 ns realisiert. Die CSU führt die Sicherung der relevanten Informationen in den task-spezifischen Kontrollblock in 100 ns durch, während das Laden der benötigten Informationen für die neue Task in 380 ns erfolgt.

Die Anzahl der Zyklen betragen für sämtliche Befehle zwischen 3 und 4 Takte. Befehle mit der direkten und der unmittelbaren Adressierung werden in 3 Takten, während Befehle mit der indirekten Adressierung in 4 Takten verarbeitet werden.

VI. Vergleich

Neben den sicherheitskritischen Aspekten lag der Fokus auf dem Ressourcenverbrauch bei der Entwicklung des Icarus Softcore-Prozessors. Tabelle 2 zeigt den Ressourcenverbrauch, im Vergleich zum Xilinx Microblaze [8], zum Xilinx Picoblaze [9] und zum Motorola M68HC05 [10], der aus der Synthese mit Xilinx ISE resultiert. Beim Vergleich muss beachtet werden, dass der Microblaze ein 32 Bit Prozessor ist, während es sich bei Icarus um ein 16 Bit Prozessor handelt. Dagegen sind der M68HC05 und der Picoblaze 8 Bit Prozessoren.

Tabelle 2: Vergleich des Ressourcenverbrauchs zwischen Icarus, Xilinx Microblaze v7.3a, Xilinx Picoblaze [11] und Motorola M68HC05 [11], auf Basis des Xilinx XC3S200-4ft256 FPGAs

	Icarus	Microblaze	Picoblaze	M68HC05
#Slices	485 (25%)	1335 (69%)	96 (5%)	211 (11%)
#FF	393 (10%)	1548 (40%)	181 (5%)	410 (11%)
#LUT	929 (24%)	2260 (58%)	76 (2%)	44 (1%)
#RAMB	7 (58%)	4 (33%)	0 (0%)	1 (8%)
#MULT	0 (0%)	3 (25%)	0 (0%)	1 (8%)
f_{max} [MHz]	91,25	56,86	120,4	52,10

Um den Microblaze und den Icarus vergleichbar zu machen, wurde die Microblaze-Architektur um einen Interrupt-Controller und um Message-Queues (Mailboxen) erweitert, da diese bereits integraler Bestandteil der Icarus-Architektur sind. Der Microblaze und die zusätzlichen Komponenten wurden mit dem Xilinx Platform Studio (XPS) generiert und synthetisiert. Die Syntheseergebnisse des Picoblaze und des M68HC05 beziehen sich auf [11].

Wie die Tabelle 2 verdeutlicht, sind die beiden 8 Bit Prozessoren Picoblaze und M68HC05 in Bezug auf den Ressourcenverbrauch miteinander vergleichbar. Im Gegensatz zu Icarus und Microblaze besitzen sie einen minimalen Ressourcenverbrauch.

Der Vergleich zwischen Icarus und Microblaze macht deutlich, dass der Icarus im Wesentlichen durch das mikroprogrammierte Steuerwerk enorme Ressourcenersparnis bietet. Allerdings führen das mikroprogrammierte Steuerwerk und die segmentierten Speicher des Icarus zu einem höheren Bedarf an RAM-Blöcken.

VII. Fazit

A. Zusammenfassung

Im vorliegenden Beitrag wurde der 16 Bit Softcore-Prozessor Icarus vorgestellt, der für sicherheitskritische Anwendungen in eingebettete Systeme konzipiert und entwickelt wurde. Dabei wurde insbesondere auf die Aspekte der taskbasierten Architektur eingegangen, die für die Umsetzung der Sicherheitskriterien von Bedeutung sind. Anhand eines Vergleiches mit dem Microblaze konnte gezeigt werden, dass die Synthese des Icarus zu erheblichen Einsparungen an Ressourcen (Slices) in FPGAs führt. Die Synthese zeigte, dass eine Taskumschaltung innerhalb von 560 ns durchgeführt werden kann und somit eine Echtzeit-Kommunikation zwischen Tasks möglich ist.

Die besonderen Merkmale des Softcore-Prozessors sind Speicherschutz durch Segmentierung und Adressierung mittels der Konkatenation eines Deskriptors mit der Adresse innerhalb eines Segments. Durch eine Speicher-Speicher-Architektur entfallen die Datenregister, sowie die dazugehörigen Befehle zum Laden und Speichern. Die Intertaskkommunikation wird über eine ITC-Matrix definiert. Mit Hilfe dieser lässt sich die Kommunikation, welche auf atomaren Befehlen beruht, zwischen zwei oder mehreren Tasks überwachen. Aufgrund des einheitlichen Message-Protokolls ist es ohne zusätzliche Handshake-Informationen möglich eine Kommunikation zwischen Daten und Referenzen zu unterscheiden.

B. Ausblick

Um den Entwickler zu unterstützen, könnte das System durch eine Debug-Schnittstelle im Compiler erweitert werden. Des Weiteren hat die Hardware-Synthese gezeigt, dass die maximale Taktrate f_{max} von 91,25 MHz des Icarus auf mehrere kritische Pfade zurückzuführen ist. Durch Optimierung der ALU und anderer Komponenten könnte eine höhere Taktrate erreicht werden.

Literaturverzeichnis

[1] Xilinx, „Using Block RAM in Spartan-3 Generation FPGAs,“ Application Note 463, *Xilinx Inc*., 1. März 2005.

[2] A. Schaaf, B. Teppert, T. Tornar, I. Schoppa, „Ressourcen-Optimierung durch Einsatz primitiver FPGA-Komponenten“, *45. Workshop der MPC-Gruppe Baden-Württemberg, Hrsg. Hochschule Ulm*, S.41-45, Febr. 2011, ISSN 1868-9221.

[3] U. Brinkschulte, T. Ungerer, „Mikrocontroller und Mikroprozessoren“, ISBN: 978-3-642-05398-6, *Springer Berlin Heidelberg*, 2010.

[4] John L. Hennessy, David A. Patterson, „Rechnerarchitektur”, ISBN: 3528051736,*Vieweg*, 1994.

[5] K. Wüst, „Mikroprozessortechnik: Grundlagen, Architekturen, Schaltungstechnik und Betrieb von Mikroprozessoren und Mikrocontrollern”, ISBN: 9783834809063, *Vieweg+Teubner,* 2010.

[6] A. Tanenbaum, „Computerarchitektur“, 5. Auflage, *Pearson Studium*, München, 2006.

[7] W. Stallings, „Betriebssysteme“, 4. Auflage, *Pearson Studium, München*, 2003.

[8] Xilinx, „MicroBlaze Processor Reference Guide“, UG081 Rev. 12, *Xilinx Inc*., 01. März. 2011.

[9] Xilinx, „PicoBlaze 8-bit Embedded Microcontroller User Guide“, UG129, *Xilinx Inc.*, 22. Juni. 2011.

[10] Motorola, „M68HC05 Microcontroller Applications Guide“, Rev. 3.0, *Motorola Inc*., Phoenix, 1998.

[11] Ch. Kielmann, D. Stengele, I. Schoppa, „Einsatz von RAM-Blöcken als mikroprogrammierte Steuerwerke in FPGAs“, *45. Workshop der MPC-Gruppe Baden-Württemberg, Hrsg. Hochschule Ulm*, S. 33-40, Febr. 2011, ISSN 1868-9221.

Kai S. Brach studierte Software Engineering an der HTWG Konstanz und erhielt dort im Jahre 2010 den akademischen Grad des B. Sc. Im Rahmen seiner Bachelor-Arbeit befasste er sich mit der Navigation und Hindernisvermeidung von mobilen Robotern. Er ist Student des Master-Studienganges Informatik, sowie wissenschaftlicher Mitarbeiter der Hochschule Konstanz.

Hannes Hennig studierte Technische Informatik an der HTWG Konstanz und erhielt dort im Jahre 2010 den akademischen Grad B. Sc. Im Rahmen seiner Bachelor-Arbeit befasste er sich mit generischem Hardware-Software Codesign. Neben seinem Master-Studium arbeitet er in der Software-Entwicklung beim Sensorhersteller SICK AG

Christian Kielmann studierte Technische Informatik an der HTWG Konstanz und erhielt dort im Jahre 2010 den akademischen Grad B. Sc. Im Rahmen einer Bachelor-Arbeit befasste er sich mit Soft-Core-Prozessoren. Neben dem Master-Studium arbeitete er in der Spezialmesstechnik beim Triebwerkshersteller Rolls-Royce Deutschland.

Kamil Wistuba erhielt den akademischen Grad des B. Sc. in Technischer Informatik im Jahr 2010 von der Hochschule Konstanz. Er ist Student des Master-Studienganges Informatik an der Hochschule Konstanz. Neben seinem Studium arbeitet er in der Software-Entwicklung bei TRW Automotive Deutschland.

Irenäus Schoppa studierte Informatik an der Technischen Universität Berlin und erhielt dort im Jahre 1993 den akademischen Grad Dipl.-Informatiker. Im Jahre 1998 promovierte er dort zum Dr.-Ing. Seit dem Jahr 2008 ist er Professor für Hardware-Software Codesign an der HTWG Konstanz.

Realization of a Volterra-Based Postdistortion Algorithm for Radar Receivers

Georg J. Vallant, Wolfgang Schlecker, Friedrich K. Jondral

***Abstract*—Modern Pulse Radar Receivers have an increasing demand for high bandwidth and dynamic range. Digitization of the analog signal at IF allows architectural benefits, but places tight requirements on the ADC performance. How high an IF can be placed is severely restricted by the ADC's SFDR roll-off with increasing input frequency. While the SNR remains quite constant and can be improved by various Radar processing gains, SFDR remains the limiting figure of merit, as spurious contributions will cause potential false targets in the radar processor. Weaker signals of interest will be masked by distortion induced by strong interferers or clutter.**

We propose a digital, system-level linearization algorithm for ADCs, which allows increased receiver SFDR at high input frequencies. The model used for identification and correction is a constrained version of the general Volterra series. We focus on the FPGA implementation, which exploits a parallel processing structure, in order to perform all computations at the necessary sample rate. The bit-true match between Matlab and VHDL code is verified in ModelSim. For estimating the hardware effort, a synthesis report for standard FPGAs is given. The application of the IP-Core in tandem with a commerical 16-Bit 120 MSPS ADC yielded 25 dB increased SFDR.

***Index Terms*—Radar, ADC, Subsampling, Volterra and Hammerstein Models, SFDR, Postdistortion, Nonlinear Distortion, Digital Enhancement, FPGA**

I. Introduction

Although the term Software Defined Radio (SDR) evolved in communication technology, refering to a flexible radio, which adapts to various protocols and waveforms with software changes only, some of the general paradigms can also be applied to Radar. The ideal software radio/radar receiver would in this case be an ADC located directly at the antenna, but for a number of reasons this is a rather opportunistic view and not possible if high performance is demanded. Fig. 1 depicts the most basic IF-Subsampling scheme with a digital downconversion to complex baseband [1]. The bandlimited, real-valued signal located at the center of a Nyquist Zone, gets sampled and shifted to 0 Hz using a complex phasor. Sampling at higher IF frequencies means relaxed requirements for the preceeding RF components (e. g. mirror rejection filters) or even a skipped 2nd mixing stage, which allows reduced noise, nonlinearity, costs and also complexity, as frequency planning is simplified in a single-stage heterodyne architecture. Ideally, the classical dual-stage heterodyne Radar receiver would become a single-stage receiver, with the performance remaining the same. While this sounds tempting, some effects need to be considered: Aperture jitter is one important point, because sampling a GHz frequency can only be achieved at the expense of a certain SNR loss.

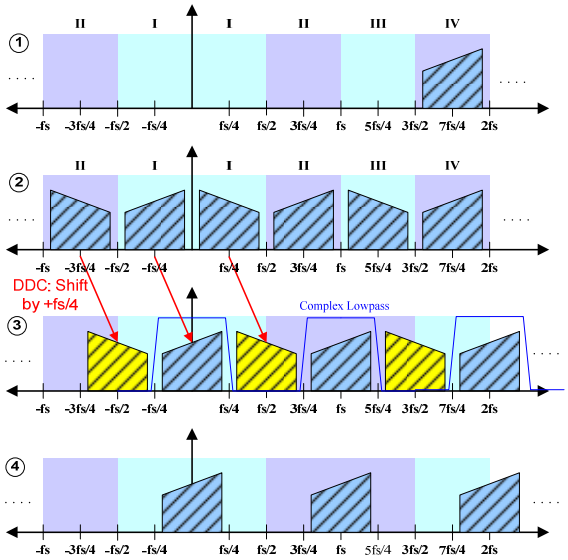


Fig. 1: Principle of IF-Subsampling and Digital Downconversion of a bandlimited IF-Signal centered at a specific Nyquist Zone

$$SNR_{jitter} = -20\log_{10}(2\pi f_{in}\sqrt{t_{ap,\mathrm{int}}^{\ 2} + t_{ap,ext}^{\ 2}})$$

Georg Vallant (georg.vallant@cassidian.com) and Wolfgang Schlecker (wolfgang.schlecker@cassidian.com) are with Cassidian Electronics, Woerthstrasse 85, D-89077 Ulm, Germany

Friedrich K. Jondral (friedrich.jondral@kit.edu) is with Karlsruhe Institute of Technology (KIT), Communications Engineering Lab, D-76128 Karlsruhe, Germany

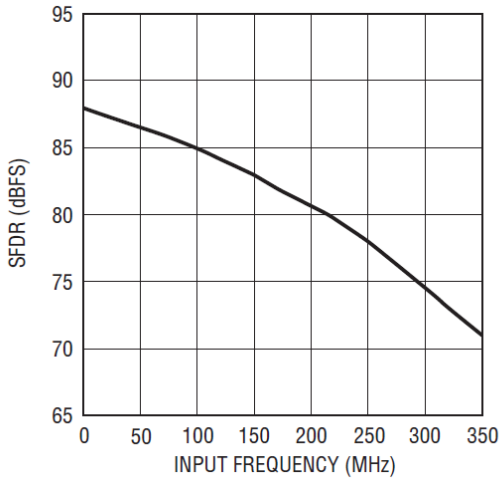


Fig. 2: Typical SFDR-rolloff due to dynamic frontend nonlinearity of pipelined ADCs. Exemplary plot taken from data sheet of LTC2262-14 (150 MSPS, 14 Bit) [5]

Today's ADCs and clock sources offer tremendously low jitter. Combined rms values as low as 100 fs are not uncommon. This allows sampling a 1 GHz full scale sinusoid with 64 dB SNR. The second point is that the linearity of ADCs severely drops as the input frequency is increased [3], a fact that sets an upper limit on the IF, as nonlinear distortion tends to degrade the detection process and thus the Radar's capability to detect weak signals of interest in the presence of strong clutter or another strong signal. Interestingly the full power bandwidth (FPBW) of most pipelined ADCs is quite high (e.g. 1 GHz), but without model-based postdistortion only some 10 - 20 % of the bandwitdth can be used for IF placement. If the notorious SFDR degradation, as depicted for convience in Fig. 2, could be tackled directly in the receiver's digital preprocessing, the step towards a single-stage receiver or relaxed analog receiver requirements becomes a real perspective.

Digital linearization of an ADC (mainly consisting of the external input circuitry, the Track-and-Hold-Input and the quantizer) demands for sophisticated nonlinear, frequency-dependent models and a good understanding of the associated error mechanisms. The advent of methods that deliver some sort of "Digital Assistance" is linked to shrinking of transistor sizes and advances in semiconductor technology [6].

While circuit-level based digital correction is very limited to a specific ADC architecture or design, a system-level algorithm could increase the performances in a more flexible way. This contribution focuses on system-level correction and has the following structure: In Chapter II we will discuss the modeling approach and the correction model, in Chapter III we will discuss the FPGA implementation alongside the synthesis report, in Chapter IV we will present results using a commercial ADC and in Chapter V we will give a conclusion.

Table 1: Hardware effort (number of coefficients) for full Volterra model and constrained Hammerstein/Wiener models up to memory length M of 6 and Nonlinearity Order L of 3

Volterra Complexity						
L/M	1	2	3	4	5	6
1 linear	1	2	3	4	5	5
2 quad	2	6	12	20	30	42
3 cubic	3	14	39	84	155	258
Hammerstein/Wiener Complexity						
L/M	1	2	3	4	5	6
1 linear	1	2	3	4	5	5
2 quad	2	4	6	8	10	12
3 cubic	3	6	9	12	15	18

II. Modeling and Correction

A. Modeling of Nonlinear, Dynamic Systems

The static, polynomial-based memoryless power series

$$y(t) = \sum_{i=0}^{L} a_i x(t)^i$$

is not able to describe frequency-dependent effects. Extending the series on basis of LTI system theory

$$y(t) = T_1[x(t)] = \int_{-\infty}^{+\infty} h_1(\tau_1) \cdot x(t-\tau_1) d\tau_1$$

yields the so-called Volterra Series [2], which is a complex model for weakly nonlinear systems with fading memory. The output of the system will then be

$$y(t) = T_1[x(t)] + T_2[x(t)] + \ldots + T_L[x(t)]$$

Signal behaviour is described by higher-order impulse responses, called Volterra kernels, and higher-order convolution integrals, called Volterra operators. Its practical use in its full form is however strongly limited, as the CC (computational complexity) goes exponentially with memory length M and model order L

$$CC_{Volterra} = \sum_{q=0}^{L} M_q^q$$

Although many electrical systems have low order nonlinearity (up to cubic) the memory lengths might still be quite high. In addition there are known problems with numerical stability and the identification (e.g. a pure 3-tone signal is needed to extract the 3rd order Volterra kernel). Note that this approximation omits any further effort due to the need for increased processing rate through interpolation, as all

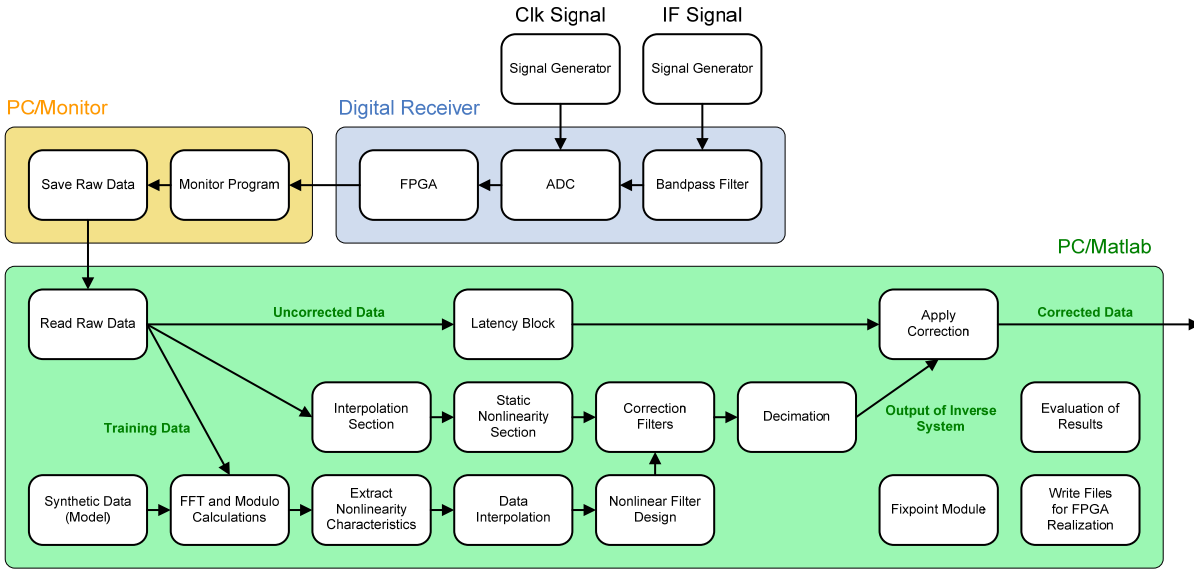


Fig. 3: Hardware/Software setup for showing improved ADC performance at high input frequencies with a Volterra-based linearizer

frequencies inside the discrete-time model must fulfil Nyquist.

But, using knowledge from ADC structure we can simplify Volterra into Hammerstein or Wiener box-models. This forces a special symmetry on IMD (intermodulation) products. Table 1 gives a quick CC comparison between both models. A polynomial Hammerstein system offers a good compromise between CC and modeling possibilities. The discrete Hammerstein model has static nonlinearities followed by linear FIR filters. It can be written as

$$y(x(k)) = \sum_{i=1}^{L} \sum_{m=0}^{M} h_i(m)(x(k-m))^i$$

ADC core nonlinarity on the other hand is defined by INL/DNL, which describes only the static transfer characteristics by defining the deviation from a perfect staircase function after correcting for gain and offset errors. INL-based correction methods are often applied, but with increased input frequency INL is not able to describe the nonlinarity seen at the output of the ADC. Refering to [3][4] we state that correcting nonlinearity behaviour of ADC's even with a frequency-dependent INL model that captures 0 and +/- 180° phase shifts of distortion terms is insufficient for high-IF sampling configurations. Thus no ADC parameter directly relates to dynamic nonlinearity, but dynamic sinusoid tests are measures of dynamic nonlinearity.

B. Model-Based Postcorrection Scheme

Injecting a set of pure sine tones, with frequencies that fulfil the requirement for coherent sampling

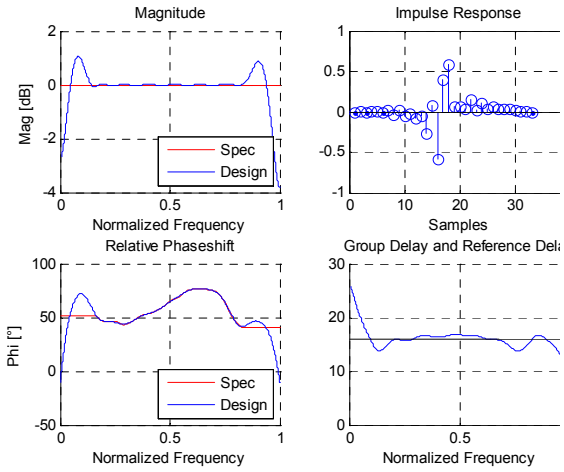


Fig. 4: Exemplary nonlinear filter design of the quadratic subfilter (phase only) as used in the correction scheme

$$f_{coherent} = \left(\frac{J}{N}\right) \cdot f_s = J \cdot d_f$$

with J being an integer prime to N, N being the Number of Samples, f_s being the sampling frequency and d_f being the DFT binwidth, allows an investigation of ADC memory effects. If N is a power of two, all odd values of J will fulfil the requirement. Then evaluated DFT spectra will have discrete lines (there is no leakage effect and need for faulty windowing) and also any phase relationships between aliased distortion terms and the fundamental can be extracted using modulo calculations.

The setup for the following investigations shown in Fig. 3 is as follows: A digital receiver encompassing an input circuitry (usually a transformer or balun for single-ended to differential conversion), the ADC

itself and an FPGA, is injected with pure, bandpass-filtered sine tones, stepped across the Nyquist zone under use and adjusted to -1 dBfs amplitude. Coherent raw data is saved from a PC with a monitor program, as it is used for typical ADC performance measurements. An extended FFT evaluation of the saved raw data set in MATLAB forms the basis for creating an inverse correction model, which consists of an interpolation section and arbitrary, nonlinear-phase FIR filters. A representative filter design procedure can be seen in Fig. 4. To tackle the ADC distortion, the polynomial correction model was extended up to 3rd order. The correction value for each sample is formed by calculating a weighted sum of sorrounding samples followed by a decimation to the original ADC sample rate. The correction model itself needs a higher processing rate to avoid any internal aliasing. The uncorrected signal itself remains unfiltered, it is delayed by a few samples, according to the overall latency inside the nonlinear correction paths. Establishing the system's inverse yields an theoretical SFDR improvement of

$$SFDR_{impr} = 10 \cdot \log 10(1 - \\ - 2(1 + \Delta A / A) \cdot \cos(\Delta\phi) + (1 + \Delta A / A)^2)$$

For example, to achieve 25 dB increase in SFDR, the inverse must match the model with a residual phase error $\Delta\phi$ of 2-3° and a gain matching $\Delta A/A$ of at least 0.25 dB [7].

Other optional modules in our setup are fixpoint processing for bit-true modeling of all quantization errors that need to be addressed in hardware, evaluation of all results and storage of the necessary FPGA-related data like filter coefficient sets and comparison signals.

III. FPGA Implementation

A. Polyphase Decomposition and Quantization

Every FIR transfer function can be written in its polyphase decomposition or parallel form. Assuming an L-branch decomposition, the general case is

$$H(z) = \sum_{m=0}^{L-1} z^{-m} \cdot E_m(z^L)$$

$$E_m(z) = \sum_{n=0}^{|(N+1)/L|} h[Ln + m] z^{-m}$$

$$h[n] = 0 \quad for\ n > N$$

where $E_m(z)$ are the subfilters and N is the original FIR filters length. This has the great advantage to perform all calculations at the necessary sample rate and exploit the FPGA's possibility for parallel processing.

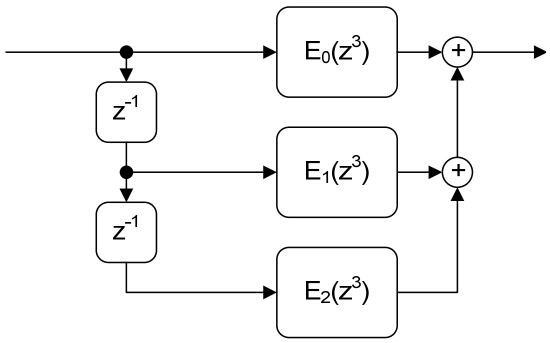


Fig. 5: 3-branch polyphase decomposition of a FIR transfer function H(z)

The chain in the nonlinearity paths is: interpolator, static nonlinearity, nonlinear-phase correction filter, and decimator. A decimator skips every L/th sample. As we do not need those samples for forming the final correction value, the corresponding filter paths can be removed from the model. Also, depending on the length of the original filter (even or odd), one needs to adjust the inner delays. Fig. 5 depicts the 3-branch case. A typical MATLAB code section describing the polyphase decomposition of a filter *b_cor3* with L=3 might be

```
b3_lp_Af=b_cor3(1:L:length(b_cor3));
b3_lp_Bf=b_cor3(2:L:length(b_cor3));
b3_lp_Cf=b_cor3(3:L:length(b_cor3));

u3_a_1=filter(b3_lp_Af,1,u3_a);
u3_b_3=filter(b3_lp_Cf,1,u3_b);
u3_c_2=filter(b3_lp_Bf,1,u3_c);

b3_lp_Bf=[0 b3_lp_Bf];
b3_lp_Cf=[0 b3_lp_Cf];

u3_a=u3_a_1+u3_c_2+u3_b_3;
```

MATLAB can be efficiently used to transform the high-level model from Fig. 3 into a bit-true hardware model. This includes fixed-point calculations inside the filters (input, filter and output quantization). The resolution inside the correction core is increased by 2-3 bits to avoid undesired overflow and quantization noise.

B. Realization and Model-Sim Verification

We realized and simulated the algorithm in a technology-independent way in plain VHDL using Mentor Graphics HDL Designer and ModelSim SE 6.5e. The detailed implementation structure is shown in Fig. 6, where the model was expanded up to cubic order. ADC raw data passes the core and is corrected by a value calculated inside the branches. Output quantization after the filter operation is done inside the Q-

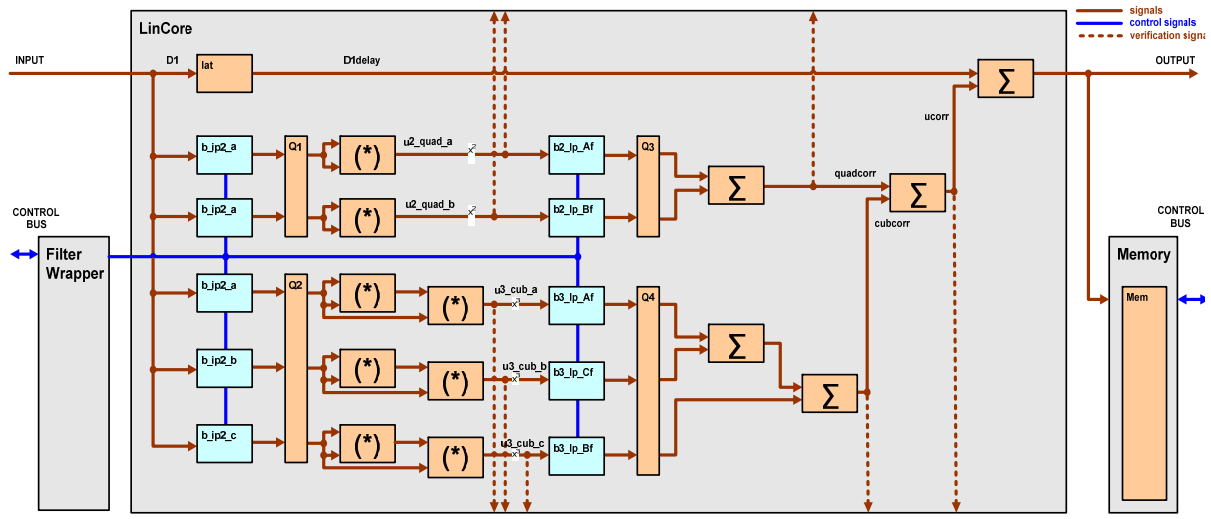


Fig. 6: Simplified VHDL Model of the Correction Core - Polyphase Realization of Correction filters

labeled blocks. A filter wrapper is used for loading filter coefficients to the interpolator and correction sections. A memory block placed after the core allows verification of improvements using the existing monitor program for readout purposes, while the corrected data stream is passed on further to the DDC (cf. Fig. 1), and afterwards in I/Q-form to the Radar processor. Dotted signals in the scheme are used for the bit-true ModelSim verification. We skipped minor latency blocks, necessary at various locations to force synchronicity before multiplicators/adders.

All parameters in the core can be configured via generics. In this case, we had 16 Bit input data, internal processing was done with 18 Bit resolution and the clock frequency was 120 MHz. In the quadratic path we used 39 taps for the interpolator and 51 taps for the correction filter. In the cubic path we used 59 taps for the interpolator and 75 taps for the correction filter. Based on our investigations we can say that memory effects of most ADC-related nonlinearity can be catched with rather moderate filter lengths in the correction section. ModelSim was used to verify the VHDL code against the bit-true MATLAB model.

Synthesis was done in Precision, Place&Route in ISE 11.3. We compared two Virtex-5 FPGAs as target platforms and took advantage of their DSP48 Elements, thus the number of used slice registers remained quite low. Table 2 shows the synthesis report: The linearization core has a realistic size in terms of computational complexity, the FPGA already located after the ADC for preprocessing tasks can also used to deploy the algorithm. As bigger sized FPGAs of the same generation (SX240) or even the smallest-sized next generation Virtex-6 have an increased Number of DSP48Es, the linearization core will be a minor contributor to overall fill grade, even if further signal processing (e. g. linear equalization to correct for time distortion of preceeding analog bandpass filters) is necessary in the preprocessing.

Table 2: Synthesis Report for two Virtex-5 FPGAs

Virtex-5 XC5VSX35T			
Slice Logic Utilization	Used	Availabl e	Utiliz ation
Number of Slice Registers	2340	21760	10%
Number of Slice LUTs	541	21760	2%
Number of occupied Slices	796	5440	14%
Number of fully used LUT FF pairs	478	2403	19%
Number of DSP48Es	156	192	81%
Virtex-5 XC5VSX240T			
Slice Logic Utilization	Used	Availabl e	Utiliz ation
Number of Slice Registers	2376	149760	1%
Number of Slice LUTs	559	149760	2%
Number of occupied Slices	853	37440	14%
Number of fully used LUT FF pairs	487	2448	19%
Number of DSP48Es	155	1056	14%

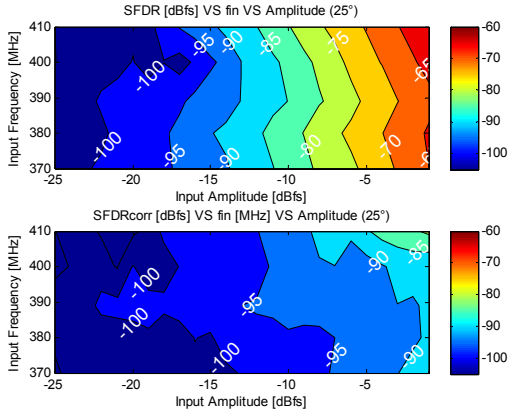


Fig. 7: SFDR Correction Results in the 7th Nyquist Zone for the LTC2208 (120 MSPS, 16 Bit), with varying input amplitudes

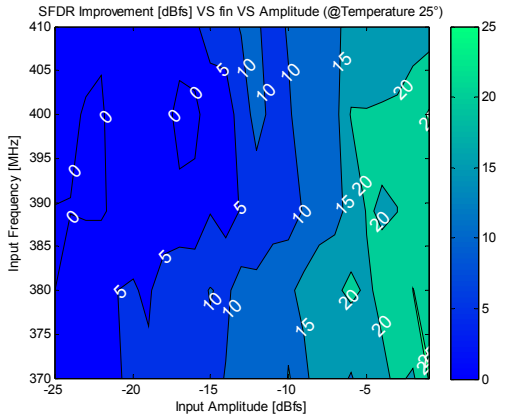


Fig. 8: SFDR relative improvement in the 7th Nyquist Zone for the LTC2208 (120 MSPS, 16 Bit), with varying input amplitudes

IV. Correction Results

We investigated the LTC2208 (120 MSPS, 16 Bit) and built an inverse correction model using the trai ning procedure described in Chapter II. At input frequencies between 370 and 410 MHz the linear attenuation of input circuit and ADC was 1 dB. The 700 MHz full power bandwidth given in the data sheet was approximately valid. We tuned the input signals, applied 21 coherently sampled and filtered singles tones with 2 MHz spacing and saved 16K of raw data each time. After the correction model was established, we evaluated the ADC-Linearizer combination at more frequency points and at various input amplitudes. Fig. 7 plots the correction results in a two-dimensional contour plot. The amplitude axis goes from -1 to -25 dBfs. At very low amplitudes, distortion terms cannot be evaluated even with a large FFT. The -100 dB results are due to the residual noise floor. At input amplitudes between -1 and -15 dB, the SFDR was sufficiently increased. The relative improvement is depicted in Fig. 8. Specification of the ADC-Linearizer combination at -1 dBfs would give a resulting SFDR of 90 dBfs and a relative SFDR improvement of 25 dB over a 40 MHz bandwidth. It can be said that distortion follows the m-th order trend known from analog signal theory. This is the major difference between dynamic nonlinearity at the frontend and the often heavily input-dependent behaviour of a static INL/HCF characteristic. For the LTC2208 static INL is very low, -100 dBfs SFDR can be achieved at low input frequencies. As the input frequency is increased, the correction core can be used to boost linearity by 20-25 dB. Of course temperature-dependent effects should not be neglected. First measurements indicate that even in a very wide temperature range improvement will stay above 10 dB. Furthermore, we measured that cubic nonlinearity for this ADC has moderate temperature dependency. We conclude that ADC temperature dependency does not nullify the merits of the ADC-Linearizer combination. One remaining point is the verification of the symmetry assumptions using a two-tone excitation at -7 dBfs. However, single-tone testing of high performance ADCs is an important first verification, as distortion terms generated after the Anti-Aliasing-Filter will be folded in-band.

V. Conclusion

Digital methods can be used to increase a receiver's spur free dynamic range. Especially ADCs used in a high-IF sampling configuration tend to have insufficient linearity. We presented a system-level correction algorithm based on Volterra series to boost the linearity of such ADCs. The algorithm uses interpolators, nonlinearities and nonlinear-phase FIR filters to establish the system's inverse. The hardware implementation takes advantage of a parallel processing structure to achieve reduced CC. We described the VHDL model and gave a synthesis report for two Virtex-5 FPGAs. The report shows that the linearization core has a realistic size for application in a digital receiver. We presented correction results by placing the core in series to a commercial ADC (16 Bit, 120 MSPS). SFDR could be increased by 25 dB in the 7th Nyquist Zone. The results are valid in the converters whole frequency-amplitude plane. In order to catch any residual temperature-dependent effects, the filter wrapper outside the core offers the possibility to reload sets of filter coefficients or even adapt the filters based on a control signal from the processing partition. This will be a topic for future studies.

ACKNOWLEDGEMENT

The authors would like to express their thanks to U. Schneider (Senior Manager Digital Equipment) and M. Schlumpp (Head of Programmable Hardware). We acknowledge M. Epp, whose technical suggestions on ADCs significantly enriched this work, and Dr. J. Dederer, who gave important system-specific hints. Finally, we thank K. Schmidt and T. Eifler for support regarding the test receiver platform.

REFERENCES

[1] M. Skolnik et .al, „Radar Handbook" *McGraw-Hill*, *3rd Edition*, 2008.

[2] M. Schetzen, „The Volterra and Wiener Theories of Nonlinear Systems" *Krieger Publishing Company*, Reprint Edition 1980/2006.

[3] G. Vallant, J. Dederer, M. Epp, W. Schlecker, and F. K. Jondral, "A Linearization Strategy for Undersampling Analog-to-Digital Converters", *IWADC2011 16th International Workshop on ADC Modelling, Testing and Design & IEEE 2011 ADC Forum,* Orvieto, Italy, 2011.

[4] H. Fraz, N. Björsell, J. Kenney, R. Sperlich, "Prediction of Harmonic Distortion in ADCs using Dynamic Integral Non-Linearity Model", *IEEE Behavioral Modeling and Simulation Workshop (BMAS 2009),* 2009.

[5] Linear Technology, "LTC2262-14 Datasheet" http://cds.linear.com/docs/Datasheet/226214fa.pdf

[6] B. Murmann, C. Vogel, H. Koppel, "Digitally Enhanced Analog Circuits: System Aspects", *Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS 2008)*, Seattle (USA), 2008, pp. 560-563.

[7] J. Vuolevi, T. Rahkonen, Distortion in RF Power Amplifiers. *Artech House*, 2003.

Georg Vallant received the B.Eng. in Electrical Engineering and M.Eng in Systems Engineering from the Hochschule Ulm in 2008 and 2010, respectively. After completing a Trainee program at CASSIDIAN in 2009, he joined the company as a researcher in 2010. He is PhD student at the Karlsruhe Institute of Technology (KIT).

Wolfgang Schlecker received his Dipl.-Ing. in Electrical Engineering and the Dr.-Ing. degree from University of Ulm in 1999 and 2006. He works at CASSIDIAN in the Programmable Hardware department as team-leader of the FPGA Group.

Friedrich K. Jondral received a diploma in mathematics and a doctoral degree in natural sciences (Dr.rer.nat.) from the Technische Universität Braunschweig (Germany) in 1975 and 1979, respectively. From 1979 to 1992 he was an employee of AEG-Telefunken (now Cassidian Electronics), Ulm (Germany). Since 1993 Dr. Jondral has been Full Professor and Director of the Communications Engineering Lab (CEL) at the Karlsruhe Institute of Technology (KIT), His current research interests are in the fields of ultra wideband communications, software defined and cognitive radio, signal analysis, pattern recognition, network capacity optimization and dynamic spectrum allocation.

Dreidimensionale Schaltungsträger auf Basis der LPKF-LDS®-Technologie

Wolfgang John

Zusammenfassung—Der Trend zu Kosten-, Gewichts- und Platzersparnis bei der Entwicklung und Fertigung mechatronischer Baugruppen ist ungebrochen und wird sich in den kommenden Jahren weiter verstärken. Daher ist die Nutzung von integrierten Technologien von immer größerer Bedeutung. Die MID-Technologie bietet hierbei aufgrund der dreidimensionalen Gestaltungsfreiheit der elektrischen/elektronischen Verschaltung ideale Lösungskonzepte zur Erfüllung dieser Anforderungen. Der vorliegende Beitrag skizziert und bewertet die wichtigsten Verfahren zur Herstellung von spritzgegossenen Schaltungsträgern und beschreibt anhand ausgewählter aktueller Serienprodukte die Vorteile, die sich durch die Nutzung der räumlichen Gestaltungsfreiheit ergeben.

***Schlüsselwörter*—Molded Interconnect Devices (MID), Spritzgegossene Schaltungstäger, Laserdirektstrukturierung (LDS), Mechatronische Baugruppen.**

I. Einleitung

Bereits in den 1970er Jahren starteten erste Unternehmen vor allem in den USA, aber auch zunehmend in Europa mit Design- und Machbarkeitsstudien zur Zusammenführung von thermoplastischer Kunststofftechnik und Strukturierungstechnologien der Leiterplattenbranche.

In Deutschland führte der Wunsch nach Lösungen branchenübergreifender Problemstellungen Anfang der 1990er Jahre zunächst zur Gründung interdisziplinärer Arbeitsgruppen und im Jahr 1993 schließlich zur Gründung der Forschungsvereinigung 3-D MID e.V. am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik (FAPS) der Universität Erlangen-Nürnberg.

Wolfgang John, wolfgang.john@lpkf.com, ist Senior Consultant für die LPKF-LDS®-Technologie bei der LPKF Laser und Electronics AG in 30827 Garbsen, Osteriede 7, www.lpkf.com

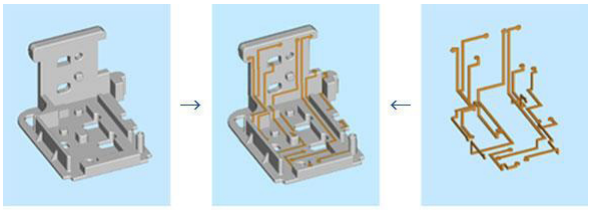


Abbildung 1: Schematische Darstellung spritzgegossener Schaltungsträger (3-D MID) [1]

Inzwischen ist dieses Netzwerk weltweit einzigartig, es besteht aus mehr als 80 Instituten und Firmen und ist eine ideale Plattform für den Einstieg und die Weiterentwicklung der MID-Technologie.

II. Was ist die MID-Technologie?

MID‘s (molded interconnect devices) sind spritzgegossene Kunststoff-Formkörper mit strukturierter Metallisierung. Man spricht häufig auch von spritzgegossenen Schaltungsträgern. Damit kombiniert diese Fertigungstechnologie die nahezu beliebige Gestaltungsfreiheit des Kunststoff-Spritzguss-Verfahrens und der damit erzielbaren mechanische Funktionalität einerseits mit den Möglichkeiten der Schaltungsträgererzeugung andererseits.

Das Grundanliegen der Technologie ist also zusammenfassend die Integration mechanischer und elektrisch und elektronischer Funktionen in einem kunststoffbasierten Mechatronik-Modul (Abbildung 1).

Zielführende MID-Entwicklungen müssen bereits in der Konzeptphase konsequent den Einsatz von MID-Lösungsansätzen berücksichtigen und die neuen Freiheitsgrade, die sich in der dritten Dimension bieten, auch aktiv und zielbewusst nutzen.

III. Vorteile von MID

Analysiert man die gestalterischen Vorteile, die MID-Lösungen gegenüber klassischen Aufbautechniken bieten können, so unterscheidet man zweckmäßig in Nutzenpotentiale der dreidimensionalen Anordnung und Nutzenpotentiale der MID-Leiterstrukturen (Abbildungen 2 und 3):

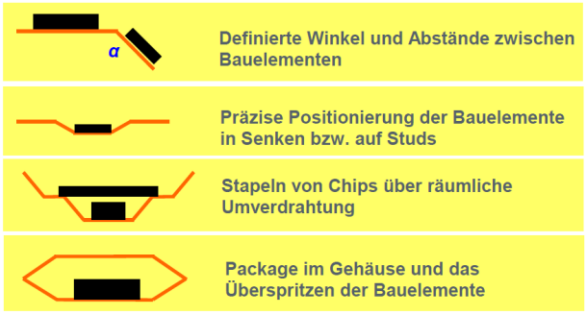


Abbildung 2: Nutzenpotentiale der 3D-Anordnung (nach [2] modifiziert)

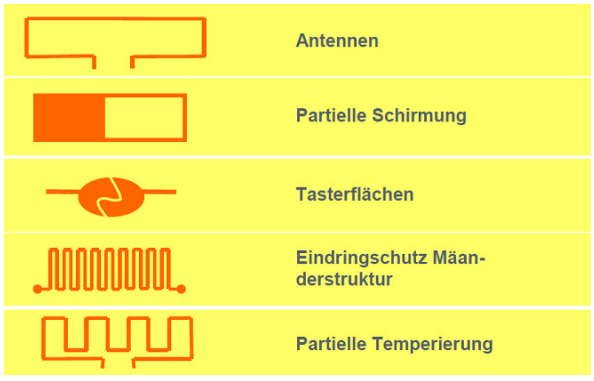


Abbildung 3: Nutzenpotentiale der MID-Leiterstrukturen (nach [2] modifiziert) zusätzlich zu Leiterbahnen und Anschlusspads, die auf Leiterplatten realisiert werden

Weitergehende, auch kommerziell nutzbare Vorteile von MID-basierten Lösungen erschließen sich oft durch eine Verkürzung der Prozesskette, eine geringere Zahl von Einzelkomponenten an der Gesamtbaugruppe und damit einhergehend eine Verringerung des Montageaufwands sowie die Erhöhung der Zuverlässigkeit.

Die Herausforderung für ein optimales MID-Design ist hierbei die gleichzeitige Berücksichtigung herstellungsspezifischer Anforderungen und die Integration der Vorteile der MID-Technologie [3].

IV. Herstellungsverfahren von MID

Die Herstellung von MID-Baugruppen unterteilt sich in die Herstellung des unbestückten Schaltungsträgers selbst und die sich anschließenden Prozesse der Aufbau- und Verbindungstechnik zur Oberflächenmontage der elektronischen Komponenten (Abbildung 4).

Erster Schritt ist in der Regel die spritzgusstechnische Herstellung des Schaltungsträgers und anschließend die Metallisierung und Strukturierung bzw. umgekehrt die Strukturierung und Metallisierung auf der Schaltungsträgeroberfläche.

Für die Herstellung des unbestückten Schaltungsträgers existieren eine Vielzahl alternativer Verfahren sowie eine Reihe unterschiedlicher Verfahrensvarianten (vgl. auch [4]).

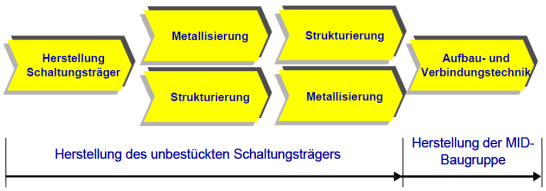


Abbildung 4: Prozesskette zur Herstellung einer bestückten MID-Baugruppe

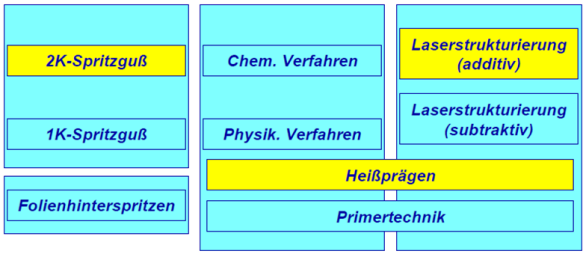


Abbildung 5: Übersicht der Herstellungsverfahren für MID-Schaltungsträger

Eine Übersicht gibt Abbildung 5 wieder, wobei die wichtigsten und praktisch relevan-ten Verfahrenstechniken gelb hervorgehoben sind.

Zu den etablierten und serienerprobten Verfahren zur Herstellung von spritzgegossenen 3D-Schaltungsträgern zählen das Zweikomponentenspritzgießen, das Heißprägen und die Laserdirektstrukturierung, wobei das letztgenannte Verfahren die jüngste Technologie mit den höchsten Zuwachsraten der letzten 5 Jahre ist. Das Auswahlverfahren der Herstellungstechnologie für ein konkretes MID-Projekt ist eine wichtige Entwicklungsstufe und erfordert ein hohes interdisziplinäres Produkt- und Prozessverständnis. Im Folgenden werden die drei wichtigsten Verfahren kurz näher erläutert und letztlich bewertet.

A. Zweikomponentenspritzguss

Bei diesem, kurz auch 2K-Verfahren genannten Prozess wird der Schaltungsträger in zwei aufeinander folgenden Spritzgussvorgängen aus unterschiedlichen Kunststoffkomponenten gefertigt. Der eine Kunststoff ist dabei metallisierbar, der andere nicht. Auf diese Weise erfolgt die Festlegung der Leiterbahnstrukturen bereits während des Spritzgussprozesses. Im Anschluss erfolgt die Metallisierung, in der Regel in chemischen Kupferbädern und mit einer anschließenden Finishbeschichtung mit Nickel und Gold.

Abbildung 6: Prozessstufen eines 2K-MID (Bildquelle: Ticona GmbH)

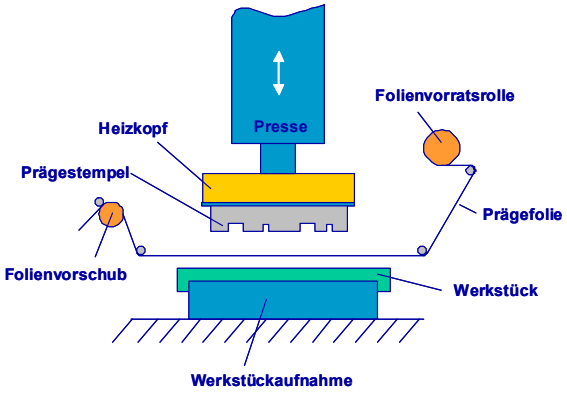


Abbildung 7: Prinzipdarstellung des Heißprägevorgangs

Abbildung 6 zeigt Beispiele der drei Prozessstufen (links: erster Schuss – nicht metallisierbare Komponente; oben mittig: zweiter Schuss – metallisierbare Kompo-nente (hell); rechts: mit Cu/Ni/Au metallisiertes MID).

Mit diesem Verfahren lassen sich je nach Fließweglängen Strukturen mit einem Line/Space von ca. 250 µm herstellen.

Beim Zweikomponentenspritzguss basiert das Schaltungslayout auf der Geometrie des Spritzgusswerkzeugs, was die Änderungshäufigkeit und –geschwindigkeit schon aus wirtschaftlichen Erwägungen begrenzt. Das Verfahren ist aus diesem Grunde prädestiniert für die Fertigung hoher Stückzahlen mit gleichbleibendem Layout. Andererseits bietet das Verfahren MID-Designern in der Entwicklungsphase den höchsten Grad an geometrischer Gestaltungsfreiheit für das Schaltungslayout.

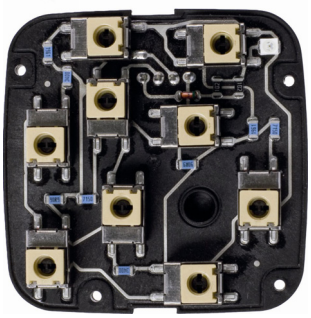


Abbildung 8: Beispiel einer heißgeprägten und bestückten MID-Baugruppe (Bildquelle: 2E mechatronic GmbH & Co. KG)

Abbildung 9: Prozesskette des LDS-Verfahrens (Erläuterungen siehe Text)

B. Heißprägen

Beim Heißprägen erfolgt die Übertragung des Schaltungslayouts auf das thermoplastische Substrat durch Ausscheren und gleichzeitiges Aufprägen aus einer Kupferfolie mittels eines beheizten Stempels, der das Leiterbild erhaben darstellt (siehe Abbildungen 7 und 8).

Mit dem Aufprägen des Leiterbildes erübrigt sich die nasschemische Metallisierung.

Als Prägefolie wird eine spezielle, galvanisch hergestellte Folie mit einem aufgerauten Rückseitentreatment verwendet, welches eine besonders hohe Haftung der Folie auf dem Kunststoff fördert. Entscheidend für die Qualität eines heißgeprägten MID-Teils ist die Auswahl der Verarbeitungsparameter Prägedruck und Temperatur des Prägestempels in Relation zum verwendeten Kunststoff. Die Palette der prägbaren Kunststoffe ist groß und reicht von Low-Cost-Kunststoffen bis zu Hochleistungswerkstoffen. Einschränkungen besitzt das Verfahren in der 3D-Fähigkeit und im Verschleiß des Prägewerkzeugs.

C. Laserdirektstrukturierung (LDS)

Das Verfahren mit dem größten Wachstumspotential und den meisten bekannt gewordenen MID-Applikationen in den letzten Jahren ist das LPKF-LDS®-Verfahren.

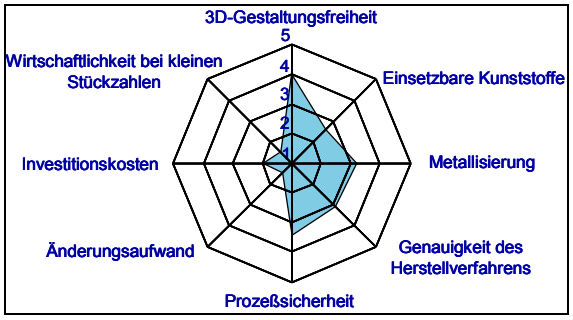


Abbildung 10: Netzdiagramm ausgewählter Merkmale für den Zweikomponentenspritzguss

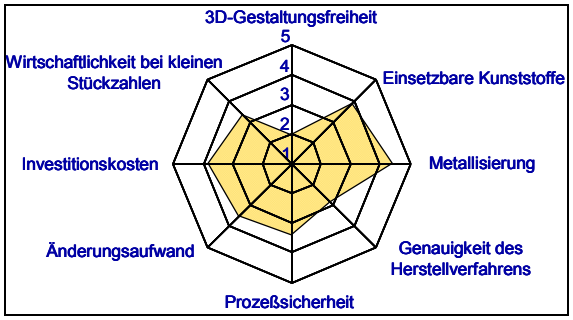


Abbildung 11: Netzdiagramm ausgewählter Merkmale für das Heißprägen

Das Verfahren verbindet die Flexibilität des Lasers mit einer kurzen, nur dreistufigen Prozesskette:

Im ersten Schritt (Abb. 9 links) erfolgt der Spritzguss des Schaltungsträgers in einem Einkomponenten-Spritzgusswerkzeug aus einem mit einem speziellen Additiv dotierten Kunststoff. Der Branche steht heute ein breites Spektrum LDS-fähiger Kunststoffe, vom Commodity-Kunststoff bis zum Engineering-Kunststoff zur Verfügung.

Im zweiten Prozessschritt (Bildmitte) wird mittels eines hochpräzise geführten Infrarotlasers das Schaltungsbild auf die dreidimensionale Oberfläche des Kunststoffteils gescannt. Da wo der Laserstrahl auf die Oberfläche des dotierten Kunststoffs trifft, wird er mikroaufgeraut und das Additiv wandelt sich unter dem Einfluss der Laserenergie in metallhaltige Keime, die als Katalysatoren für die chemische Verkupferung im anschließenden Schritt dienen.

Im dritten und letzten Prozessschritt (Abb. 8 rechts) werden die laserstrukturierten MID-Teile zunächst chemisch verkupfert und anschließend in der Regel mit Finishschichten aus Nickel und schließlich Flashgold metallisiert.

Neben der besonders guten Eignung des LDS-Verfahrens zur Feinleiterstrukturierung zeichnet sich das Verfahren durch eine hohe Flexibilität aus, da Änderungen im Schaltungslayout lediglich über Änderungen in der Software des Strukturierungslasers realisiert werden können.

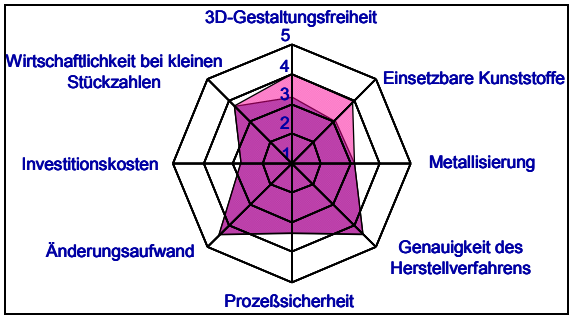


Abbildung 12: Netzdiagramm ausgewählter Merkmale für das LDS-Verfahren

Änderungen von Werkzeugen, so wie sie beim Zweikomponentenspritzguss bzw. beim Heißprägen notwendig sind, entfallen hier völlig. Insofern ist das LDS-Verfahren prädestiniert für die Fertigung von MID-Baugruppen mit hoher Varianten-vielfalt. Ein weiterer für den Automotive-Bereich wichtiger Aspekt ist die Tatsache, dass mit dem Laser auch ein fortlaufender DataMatrix-Code zur Rückver-folgbarkeit der Baugruppe strukturiert werden kann.

D. Bewertung der Herstellverfahren

Wie bereits erwähnt, ist das Auswahlverfahren der Herstelltechnologie in einem konkreten MID-Projekt ein wichtiger Schritt im Entwicklungsprozess. Die hier näher betrachteten drei Herstellverfahren lassen sich übersichtlich am zweckmäßigsten in einem Netzdiagramm vergleichen, wenn man einige wichtige technologische und fertigungsrelevante Kriterien auf den Achsen eines solchen Diagramms auf einer Skala von 1 – 5 wertet (wobei 1 eine Schwäche und 5 eine Stärke des jeweiligen Merkmals darstellt) und schließlich die Größen der sich durch Verbinden der Wertungspunkte innerhalb eines Diagramms für die drei Verfahrenstechniken vergleicht [5].

In den Abbildungen 10 – 12 sind die Diagramme der jeweiligen Verfahren abgebildet.

Die Stärke des Zweikomponentenspritzgussverfahrens liegt in der 3D-Gestaltungsfreiheit, Schwächen besitzt das Verfahren vor allem in den relativ hohen Investitionskosten für das 2K-Spritzgusswerkzeug und bei notwendigen Änderungen im Schaltungslayout, die notwendigerweise teure und zeitaufwändige Änderungen des Spritzgiesswerkzeugs nach sich ziehen.

Das in Abbildung 11 gezeigte Netzdiagramm für das Heißprägen zeigt einen sehr deutlichen Vorteil in der Metallisierung, da sich bei diesem Verfahren eine nasschemische Metallisierung erübrigt. Sehr eingeschränkt ist dagegen die 3D-Gestaltungsfreiheit.

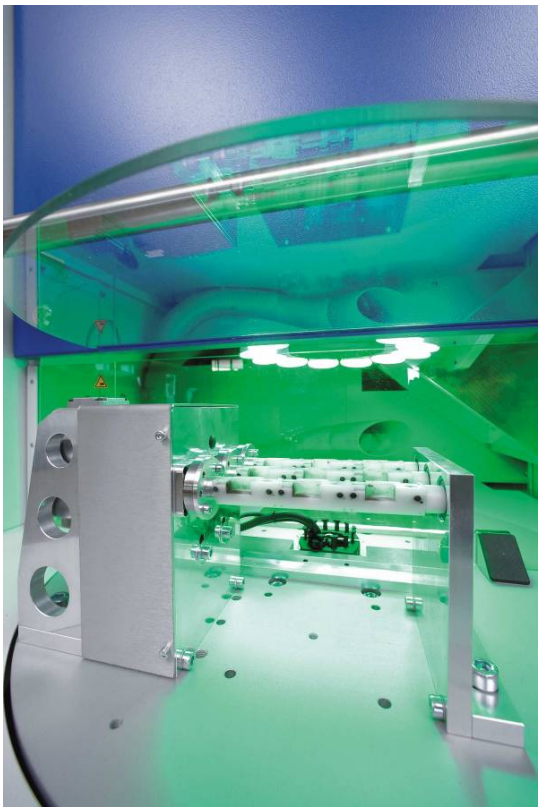


Abbildung 13: Strukturierungsraum des Laserstrukturierers LPKF MicroLine 3D mit einem Laserkopf

Abbildung 12 zeigt schließlich das Netzdiagramm ausgewählter Merkmale für das Laserdirektstrukturieren mit der Besonderheit, dass im Vergleich zur dunkelroten Fläche mit der hellroten Fläche ein Zuwachs der 3D-Gestaltungsfreiheit und der einsetzbaren Kunststoffe verbunden ist. Das erklärt sich damit, dass die dunkelrote Fläche für den Laserstrukturierer der Fa. LPKF vom Typ „MicroLine 3D 160“ (Abbildung 13) galt und noch heute gilt, der mit nur einem Laserkopf arbeitet.

Das bedingt das sequenzielle Strukturieren eines dreidimensionalen MID in mehreren Positionen und die Drehung des Teils in die nächste Strukturierposition kostet Zeit. Mit der Einführung einer neuen Generation von Laserstrukturierern vom Typ „LPKF FUSION 3D“ (Abbildung 14), die mit 3 – 4 Laserköpfen um das Teil herum angeordnet sind, kann parallel strukturiert werden. Das führt zu einem enormen Geschwindigkeitszuwachs beim Laserstrukturieren und erhöht so ebenfalls die 3D-Gestaltungsfreiheit.

Auf solchen Maschinen werden in Asien zum Beispiel Mobil- und Smartphone-Antennen in zweistelligen Millionenstückzahlen pro Monat strukturiert.

Gleichzeitig ist in den letzten 3 – 4 Jahren das Material-Portfolio der für den LDS-Prozess geeigneten Materialien stark ausgeweitet worden. Das rechtfertigt in der Summe die Erweiterung der Überdeckungsfläche um den hellroten Bereich und das Merkmalspektrum stellt sich nun insgesamt sehr ausgewogen dar.

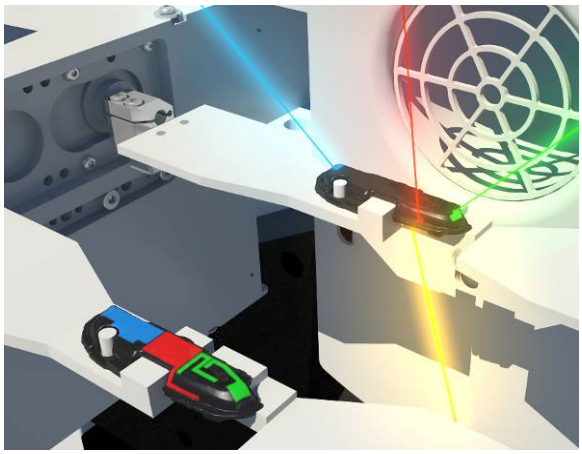


Abbildung 14: Bis zu vier Laserköpfe in sieben möglichen Positionen erhöhen erheblich die Geschwindigkeit des Laserstrukturierens mit dem LPKF Fusion 3D (Laserstrahlen und die Strukturierungsflächen schematisiert in unterschiedlichen Farben dargestellt)

V. BEISPIELE AKTUELLER SERIEN-MID

An zwei Beispielen aus dem Automotive-Bereich und einem Beispiel aus der Gebäudeklima-Technik sollen im Folgenden die Vorteile dargestellt werden, die sich aus der Nutzung der dreidimensionalen Schaltungstechnik ergeben.

A. Drucksensor für das sicherheitsrelevante ESP

Der Wertschöpfungsanteil der Elektronik an einem Kraftfahrzeug liegt derzeit bei ca. 30 %, ein Anstieg auf bis zu 40 % im Jahr 2020 wird prognostiziert [6, 7]. Durch den zunehmenden Anteil mechatronischer Systeme und wegen des begrenzten Einbauraums steigt der Druck zur Miniaturisierung.

Die MID-Technologie bietet hierbei aufgrund der hohen Gestaltungsfreiheit ideale Lösungskonzepte zur Erfüllung der genannten Anforderungen.

Abbildung 15 zeigt einen Drucksensor der neuesten Generation DS8, der von BOSCH für den Einsatz im sicherheitsrelevanten ESP entwickelt wurde und der inzwischen in hohen Stückzahlen in Serie produziert wird.

Gegenüber der Vorgängerversion zeichnet sich der Sensor durch eine deutliche Volumenreduzierung aus. Dies war nur durch die konsequente Nutzung der räumlichen Gestaltungsfreiheit, die MID-Technologien bieten können, möglich. Das 2K-MID dient als Umverdrahtung für eine hochdichte, klassisch SMD-bestückte Elektronik-Leiterplatte und stellt die elektrische Verbindung zum Druckaufnehmer auf der einen Seite und die kundenspezifischen Anschlüsse zum Steuergerät auf der anderen Seite her.

Abbildung 15: Drucksensor in 2K-Technik für ESP (Bildquelle: BOSCH)

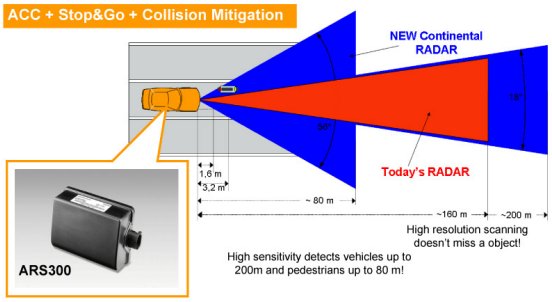


Abbildung 16: Abstandsradar ARS300 der A.D.C. Systems GmbH (Grafikquelle: [8])

B. Multifunktionsteil für Abstandsradar

Beim Abstandsradarsystem ARS300 der Fa. Continental (Abbildung 16) stand vor den Konstrukteuren des Gesamtsystems die schwierige Aufgabe, eine äußerst Platz sparenden Bauweise zu realisieren und sie kamen zu dem Schluss, dass diese Aufgabe nur mittels MID-Technik zu lösen ist.

In der Motor-Walzenantenneneinheit des Geräts ist ein MID-Teil, hergestellt im LDS-Verfahren, eingebaut, welches die komplexen Funktionen:

- Elektrische Zuführung für Motor
- Träger für Rotor-Lagedetektorelektronik
- Mechanische Lagerung der Walzenantenne auf der Motorseite im Geräteträger

übernimmt. Das MID ermöglicht damit eine kompakte Anordnung der Hauptelektronik und der Motor-Walzenantenneneinheit innerhalb des Geräts. Unter Verzicht auf manuelle Prozesse der Kabel- und Steckeranbindung wird eine automatisierte Montage erreicht (Abbildungen 17 und 18).

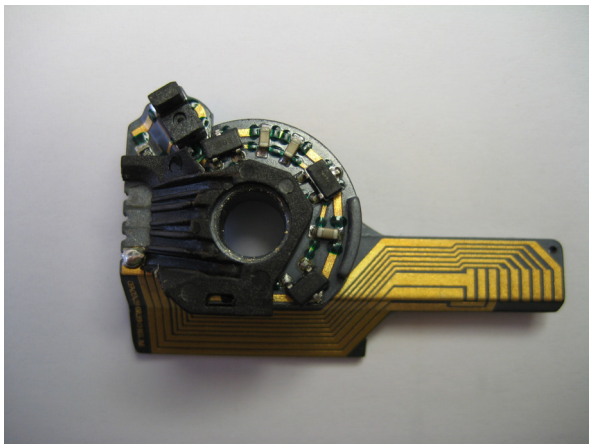


Abbildung 17: Bestücktes MID für Motor-Walzeneinheit (Bildquelle: A.D.C. Systems GmbH)

Abbildung 18: Zwischenmontierte Motor-Walzeneinheit (Bildquelle: A.D.C. Systems GmbH)

C. Strömungssensor

Miniaturisierung, weniger Einzelteile, weniger Montageaufwand und damit geringere Kosten waren die Motivation zur Weiterentwicklung eines Differenzdrucksensors für die Gebäudeklimatisierung in LDS-Technik, der in Abbildung 19 im absoluten und im Größenvergleich zur existierenden Lösung zu sehen ist. Unter Ausnutzung einer dreidimensionalen Umverdrahtung wurde im Chip-on-MID-Verfahren die aktive Chipkomponente platziert und gebondet und mit Globe-Top geschützt.

VI. FAZIT

Dreidimensionale Baugruppen bieten enormes Potential zur Optimierung von Produktionsprozessen und von Gesamtsystemen mechatronischer Baugruppen. Die Herausforderung für ein optimales MID-Design ist hierbei die gleichzeitige Berücksichtigung herstellungsspezifischer Anforderungen und die Integration der Vorteile der MID-Technologie.

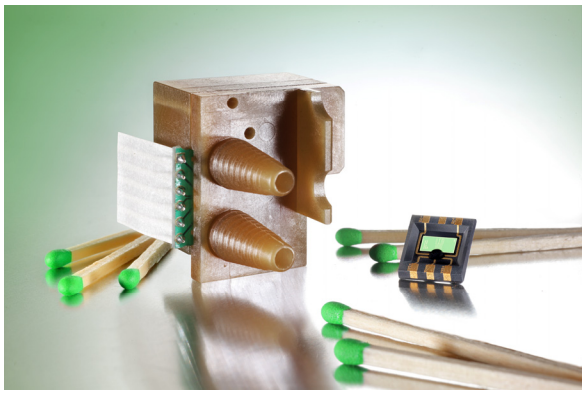


Abbildung 19: Strömungssensor für Gebäudeklimatisierung (Bildquelle: 2E mechatronic GmbH & Co. KG)

Innovative Serien-applikationen in unterschiedlichen Produktfeldern verdeutlichen die Fortschritte im Bereich der Struktu-rierung, der Metallisierung, der Aufbau- und Verbin-dungstechnik, aber auch der Kunststoffe. Insbesondere durch das LPKF-LDS-Verfahren hat die MID-Technologie allgemein in den letzten Jahren einen deutlichen Schub erfahren.

LITERATURVERZEICHNIS

[1] Forschungsvereinigung Räumliche Elektronische Baugruppen 3-D MID e.V.; www.3d-mid.de.

[2] D. Moser, J. Krause, "3D-MID – Multifunctional Packages for Sensors in Automotive Applications, Advanced Microsystems for Automotive Applications 2006" *Springer*, Berlin - Heidelberg, 2006.

[3] W. John, „Fertigungsgerechte Konstruktion dreidimensionaler Schaltungsträger" *Eugen G. Leuze PLUS Journal*, Juni 2008.

[4] Forschungsvereinigung Räumliche Elektronische Baugruppen 3-D MID e.V., „3-D MID Technologie: Räumliche Elektronische Baugruppen; Herstellungsverfahren, Gebrauchsanforderungen, Materialkennwerte" *Hanser*, München, 2004.

[5] Forschungsvereinigung Räumliche Elektronische Baugruppen 3-D MID e.V., „Chancen und Grenzen für den Einsatz der Technologie MID, Markt- und Technologieanalyse" *Heinz Nixdorf Institut der Universität Paderborn*, 2003.

[6] I. Kriebitzsch, „15 Jahre Forschung und Anwendung der MID-Technologie bei BMW" *MID-Academy Vortrag*, Stuttgart, Dezember 2007.

[7] I. Kriebitzsch, „Anforderungen an MID-Baugruppen in der automobilen Umgebung" *Eugen G. Leuze PLUS Journal,* November 2011.

[8] A. Brand, „Developing Technical and Economical Potentials in MID with Successful Serial Products" *Proceedings 8th International Congress Molded Interconnect Devices*, Nürnberg - Fürth, September 2008.

Wolfgang John ist gebürtiger Thüringer. Er studierte Chemie in Halle/Saale und promovierte über organische Halbleiter an der TH für Chemie in Merseburg. Mitte der 1990er wandte er sich dem Thema MID zu, das ihn seither auf seinem beruflichen Werdegang begleitet hat. Seit Ende 2006 ist er als Senior Consultant für die LDS-Technologie bei der LPKF Laser & Electronics AG in Garbsen tätig. Gleichzeitig ist er Vorstandsmitglied der Forschungsvereinigung 3-D MID e.V. am Lehrstuhl FAPS der Universität Erlangen – Nürnberg.

Robert Bosch Zentrum für Leistungselektronik
Hochschule Reutlingen . Universität Stuttgart . Robert Bosch GmbH

EMC Optimization of Linear Regulators with a Charge Pump – Design, Simulation and Measurements

Juergen Wittmann, Bernhard Wicht

***Abstract*— The charge pump belongs to the most critical blocks for electromagnetic compatibility (EMC) of low dropout linear regulators (LDO) because of its switching nature. The goal of this paper is to contribute charge pump design practice and a prediction method for the LDO EMC performance already in an early design phase. LDO noise coupling mechanisms are analyzed. EMC aware circuit design includes the choice of low noise architectures, the right switching frequency and noise filtering. The derived simulation method shows very good matching with EMC test results for an LDO with two different charge pumps fabricated in 350nm high-voltage BiCMOS technology. For a realistic prediction of the EMC noise magnitude, a relatively simple simulation setup gives results with <3dBµV accuracy. A tripler current mode charge pump turned out to be well suitable for EMC. Conducted emissions could be predicted and confirmed to be improved by ~50dBµV versus a conventional voltage mode charge pump.**

I. Introduction

Electronic systems are gaining more and more complexity, combining highly sensible sensory systems or receivers with complex driver stages and power management. Various circuit blocks are connected together sharing power supplies, ground lines and are interacting with each other via sensible wires as illustrated in Fig. 1. Interference between applications is the consequence, resulting in malfunction or even damages caused by coupled or conducted noise, voltage pulses in wires or by RF injection [1-5]. This is in particular critical in safety related applications like medical, automotive, or avionic. In a car for example, noise coupling into a car radio application could lead to an annoying audio noise or even to highly hazardous safety issues if breaking or airbag applications are affected. Dedicated standards for electromagnetic compatibility (EMC) [5][7] are ensured by compliance tests and measurements. On the one hand it has to be guaranteed that the circuits are robust against injected noise (e.g. direct power injection DPI), voltage pulses (e.g. ISO standard pulses) and electro static discharge (ESD). On the other hand the produced noise in a system has to be kept at a minimum, which is ensured by emissions measurements. Emissions can be distinguished between conducted and radiated emissions. Conducted emissions are measured directly at the battery and ground cables while radiated emissions are detected with an antenna [5][7].

Juergen Wittmann, juergen.wittmann@reutlingen-university.de, and Bernhard Wicht, bernhard.wicht@reutlingen-university.de, are members of the Robert Bosch Center for Power Electronics, Reutlingen University, Alteburgstraße 150, 72762 Reutlingen

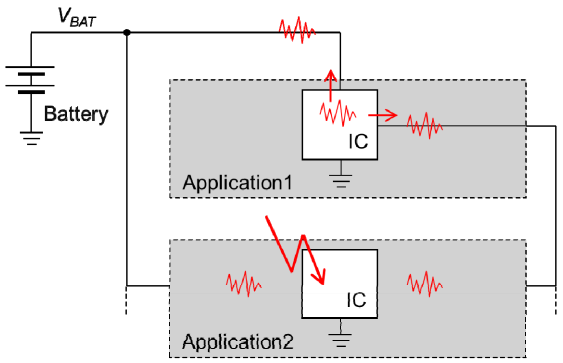


Figure 1: EMC Emissions and interference in a system.

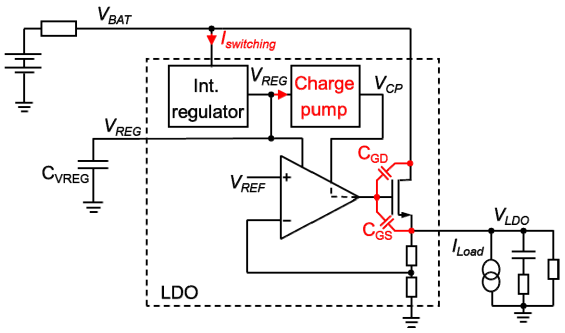


Figure 2: Block diagram of a low dropout regulator (LDO)

This paper covers integrated circuit (IC) design of linear voltage regulators (LDOs) for low emissions. The emphasis is on conducted emissions because all emissions in an IC have the same root cause while the radiated emissions are strongly depending on the PCB layout which cannot be simulated on IC level [1][2]. Root causes of emissions and EMC standards are discussed in Sect. II. In Sections III and IV two different types of known charge pump concepts are dis-

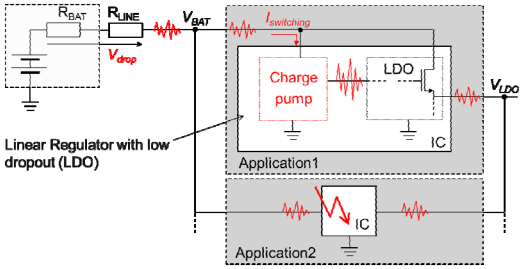


Figure 3: EMC Emissions in a system caused by a linear regulator including a charge pump.

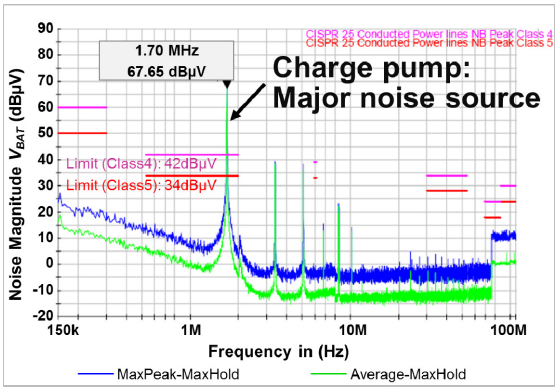


Figure 4: Measured conducted emissions with LDO exceeding the limits (upper curve: peak test, lower curve: average test).

cussed with respect to noise generation. EMC improvements by using a current controlled differential charge pump are investigated. EMC measurements are presented in Sect. V along with a simulation method suitable to predict EMC noise quantitatively already in an early design phase.

II. EMC IN LINEAR REGULATORS

In contrast to more power efficient switched mode regulators, LDOs are usually preferred for low noise supplies [8]. The combination of an n-channel power transistor and a charge pump as shown in Fig. 2 is mostly more area and cost efficient for low dropout requirements than using a p-channel transistor (see also Sect. V) [9]. An integrated charge pump is used to create the voltage overdrive for the regulator output FET from an internal supply (V_{REG}) and to supply the LDO output stage.

A charge pump is one of the critical noise sources in ICs since it is a high power switching circuit often operating at high frequencies. In order to keep the noise at a minimum, some designs activate the charge pump only if required, for instance if V_{BAT} is low. This also ensures a minimum total quiescent current in low power applications. In any case, in each charging phase large switching currents propagate through the internal regulator (charge pump input voltage V_{REG}) to the IC supply (V_{BAT} in Fig. 2 and Fig. 3). Inductive and resistive wiring and the battery output resistance create a voltage drop corresponding to the switching currents. These voltage drops are propagated to every application connected to the battery or ground line as shown in Fig. 3 in a simplified way [1-6].

In addition to current peaks, the ripple of the charge pump output voltage (VCP) results in voltage noise coupling into the LDO and its output FET [6][8]. A buffer capacitor at the charge pump output would filter the voltage ripple but is limited due to the impact on die size and cost. Fortunately, since the output FET is usually large, its gate capacitance acts as a buffer and noise filter. On the other hand, the FET represents a source follower that transfers the remaining gate noise directly to the LDO output. Since the LDO output voltage V_{LDO} is typically used to supply noise sensitive applications, the LDO output node is also of concern for emissions on system level (see Fig. 3).

Fig. 4 shows the measured conducted emissions on the supply line V_{BAT} for a predecessor IC containing an LDO with a noisy charge pump. The charge pump current creates a high noise level on the battery line at the charge pump frequency of 1.7MHz (68dBµV) and its harmonics. The unit dBµV represents the noise magnitude in dB compared to 1µV: Noise = 20dB·log10(V_{NOISE} [µV] / 1µV). The measurement in Fig. 4 was done for the CISPR 25 standard [7] with very stringent EMC pass levels used for instance for automotive applications [1][3][5]. CISPR standards define the EMC failing criteria in relevant frequency bands and the measurement environment which includes artificial networks to mimic the electrical application environment and to provide a defined termination at test. CISPR limit classes with different pass levels are available. The horizontal bars in Fig. 4 indicate the EMC limits for Class 4 and Class 5 (CISPR 25 [7]). The LDO exceeds the noise level of both classes at least by 25dBµV and thus fails the EMC test, requiring a redesign.

III. CONVENTIONAL VOLTAGE MODE CHARGE PUMP

The LDO noise shown in Fig. 4 is caused by a commonly used Dickson type tripler charge pump drawn in Fig. 5a. Stage (1) is a voltage doubler. A capacitor C_P is connected between two diodes and two complementary switches S_0, S_1. During the first phase S_1 is closed, C_P is charged close to V_{REG} level via a diode from battery, and a current I_{CPUMP} flows through S_1 to ground. In the second phase S_0 is closed and I_{CPUMP} flows from V_{REG} (out of the battery) via S_0 to the bottom plate of C_P, shifting up the top plate potential and transferring charge into stage (2) and finally towards the output which settles at $V_{CP} \sim (3V_{REG}\text{-}3V_F)$ (with diode forward voltage V_F).

Short rise times of the control signals for S_1 and S_0 are required to keep cross conduction at a minimum. This causes large inrush currents at the beginning of each pumping phase. A filter capacitor can be placed

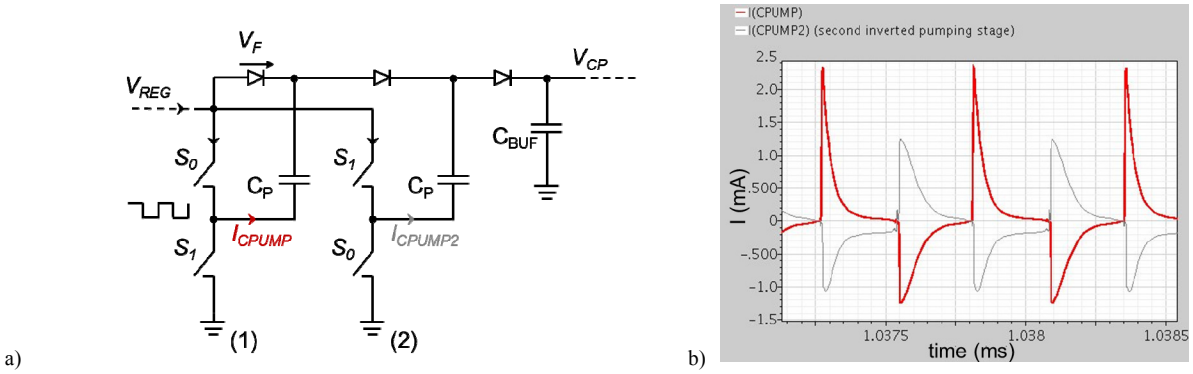


Figure 5: a) Voltage mode charge pump principle; b) Current I_{CPUMP} and I_{CPUMP2} in pumping capacitors C_P.

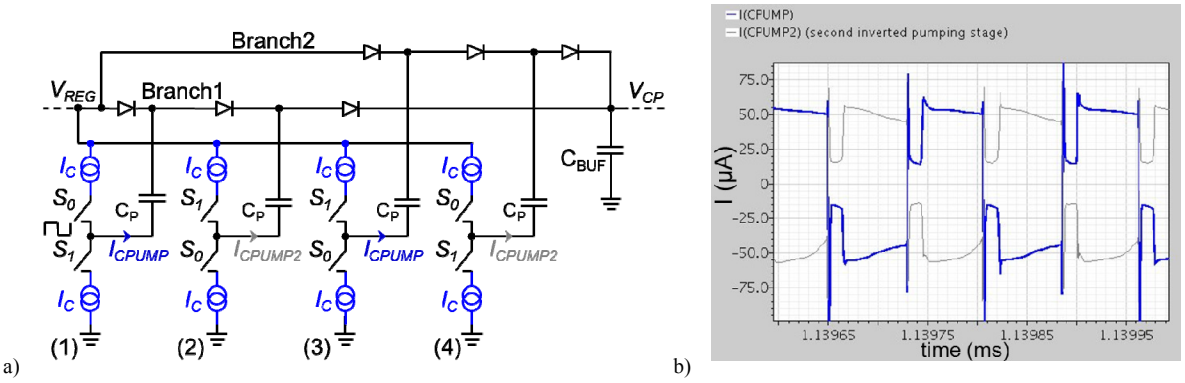


Figure 6: a) Current controlled charge pump for reduced EMC noise. b) Current I_{CPUMP} and I_{CPUMP2} in pumping capacitors C_P.

at the charge pump input (e.g. at V_{REG} in Fig. 2), improving the EMC performance at the cost of additional die area or a pin for an external capacitor.

The current I_{CPUMP} is shown in Fig. 5b (stage (1) bold curve, stage (2) grey curve). The positive currents are drawn from battery while negative currents are flowing to ground. The inrush currents in both pumping stages reach some mA and decay to zero within the same period. The associated noise will appear at battery and ground lines as well as at the charge pump output. To reduce the output voltage ripple, a large buffer capacitor C_{BUF} at the charge pump output is required to store the charge which is mainly delivered during the short current peaks. The remaining ripple contributes EMC noise to the LDO output and connected circuits.

IV. Current Controlled Differential Charge Pump Suitable for Low EMC Emissions

To improve the EMC performance a current controlled tripler charge pump with two-phase operation shown in Fig. 6 was investigated. In comparison to Fig. 5 current sources I_C have been added into the switching branches. These current sources are limiting the switching and inrush currents, in this case to I_C = 50µA [9].This leads to a constant charge current over nearly the entire pumping period. Similar to Fig. 5b, Fig. 6b (bold curve) shows the current I_{CPUMP} in stage (1) over several pumping phases. The currents are nearly constant at I_C ~50µA without any major peaks. Depending on the output load current of the charge pump the charging current within one period is switched on and off in a pulse width modulated fashion. The average current is equal to the output current. If the load current is less than I_C the charge on C_P is not fully dissipated within one period. In this case, the current sources get saturated and turn off during the charging period. By adding a parallel charge pump branch (stages (3), (4) in Fig. 6a), a fully symmetrical battery, ground and output current can be obtained in the two pumping phases. The grey curve in Fig. 6b represents the complementary current in stage (3). The superimposed branch currents result in an almost constant net current at the input. The total drawn current of $2I_C$ ~100µA is approximately constant over time on battery and ground. Similar considerations apply for the output. A significant reduction of the current peaks and thus the created noise can be achieved.

An additional improvement can be obtained by avoiding an overdesign of the driving capability of the charge pump, matching I_C with the maximum expected charge pump load current. Further, C_P and the pumping frequency f need to fulfill the condition $1/f = (C_P \cdot V_{REG})/I_C$. This way the remaining switching on and off of the pumping currents is reduced to a minimum.

As a significant advantage of this type of charge pump compared to the conventional implementation is the degree of freedom regarding the buffer capacitor

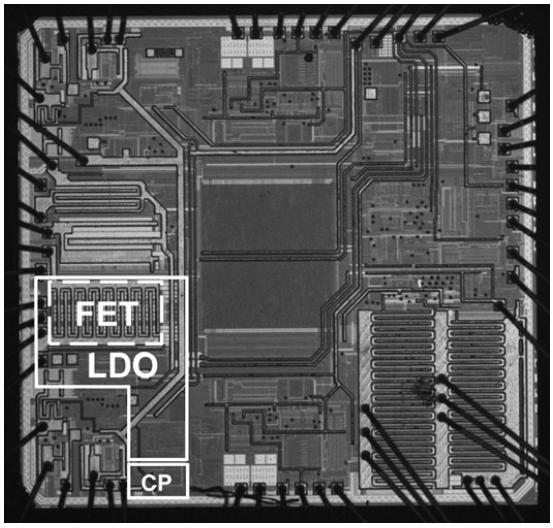


Figure 7: Microphotograph of the power supply IC with LDO and charge pump.

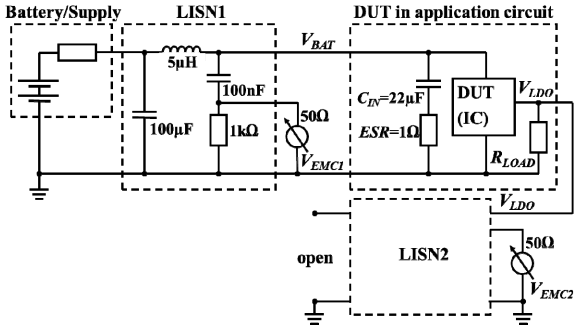


Figure 8: Setup for EMC conducted emissions test and simulation.

C_{BUF} at the output. C_{BUF} at the output can be reduced to achieve the same output voltage ripple as for the conventional pump of Fig. 5. Alternatively, keeping the capacitor the same size as in Fig. 5a the output ripple can be filtered to a very low level.

Another design criterion for the frequency selection is derived from the limit ranges of the EMC standard. The pumping frequency can be chosen in the frequency range with more relaxed limits as the increased headroom reduces the risk to fail EMC testing. Based on the CISPR 25 limits of Fig. 4 the design goal is the frequency band of 3-9MHz.

V. Measurement and Simulation

Both LDOs including the charge pump of Fig. 5 and Fig. 6, respectively, were fabricated in a 350nm high-voltage BiCMOS technology. Fig. 7 shows the microphotograph of the current-controlled charge pump as part of a multi-rail power supply IC with x-y die dimensions of ~3.5mm. Considering the size of the n-channel power FET it is obviously, that the combination of a charge pump and n-channel power FET is significantly smaller than a p-channel-LDO. A p-channel power transistor would be more than twice the area of the n-channel transistor.

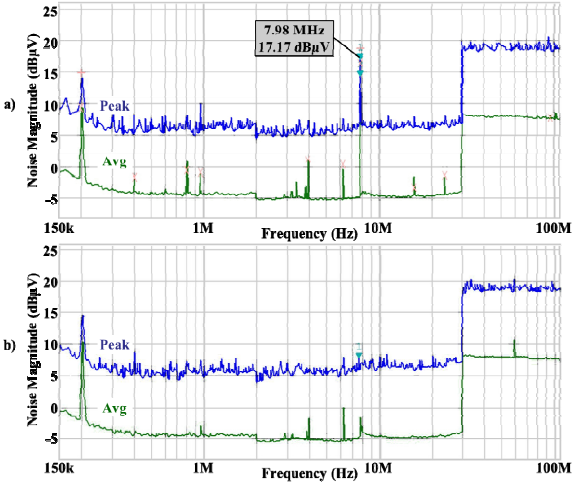


Figure 9: Measured conducted emissions of EMC improved LDO and charge pump; a) at input supply (battery); b) at LDO output.

Fig. 8 depicts the proposed setup for experimental EMC investigation of LDOs for conducted emissions following the CISPR 25 standard [7]. The challenge of any EMC test setup is to re-build a realistic environment of the specific application, to have a defined load for reproducible and comparable results and to define a coupling path of the signal to the EMC receiver. Fig. 8 indicates that the device under test (DUT), i.e. the LDO, is fully biased with all external components like in the final application, including C_{IN} and the output load [1]. A line impedance stabilization network (LISN) is used between battery and the DUT. The inductor in the LISN reproduces the wiring of the supply network of a system, e.g. the vehicle electrical supply system. A filter capacitor (100µF) eliminates unwanted noise from the supply. An RC network (100nF/1kΩ) provides the defined noise signal path to the 50Ω input of the EMC meter ($V_{EMC1,2}$). A single point ground is used, based on the assumption that in case of a vehicle the ground is directly connected to the car chassis.

The setup for the LDO output is not defined by an EMC standard since it usually supplies other IC with negligible wiring. The proposed EMC test setup of Fig. 8 utilizes a second standardized LISN (LISN2) to provide a defined termination and measurement path. The load (R_{LOAD}) is connected directly to the LDO output while the input of the LISN is left open. For the case that the LDO is loaded by another application via a long wire, the load could be applied to the input of LISN2 instead of the LDO output.

The results of Fig. 4 were measured with the setup from Fig. 8. Likewise, Fig. 9 contains the results for the EMC improved LDO according to Fig. 6, measured at the battery line and the LDO output, respectively. Figures 3 and 9 both show the noise magnitude for peak (upper curve) and average (lower curve) test.

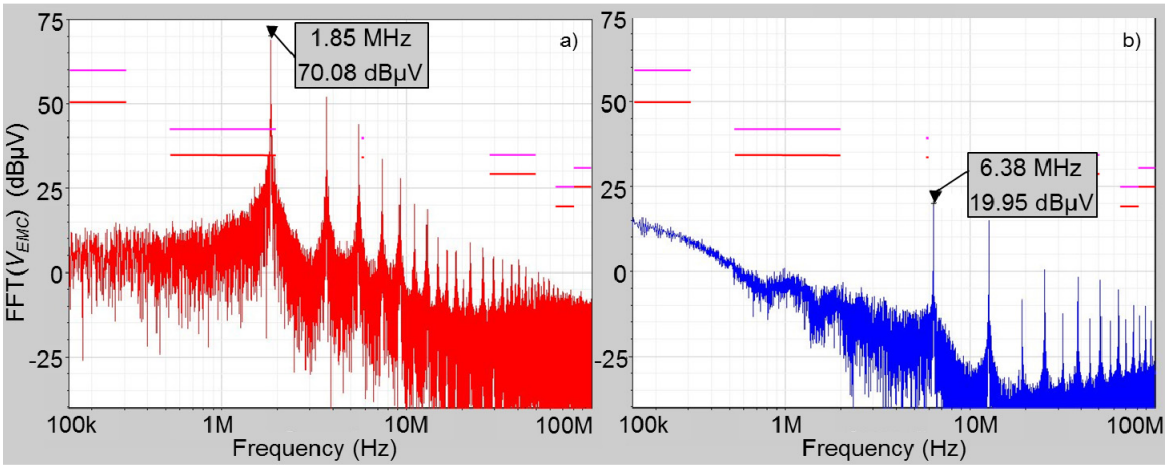


Figure 10: EMC simulation results: a) Original LDO design. b) LDO design with improved charge pump.

In Fig. 9a, a noise peak at a frequency of ~8MHz occurs with a magnitude of ~17dBµV (peak) and ~14dBµV (average). This is an improvement of more than 50dBµV compared to the conventional design (Fig. 5). Both designs use C_{BUF} ~10pF at the charge pump output. The frequency of the peak correlates to the charge pump oscillator frequency. Fig. 9b proofs that there is only very low remaining noise at the LDO output for the charge pump concept of Fig. 6.

A challenge during the design is the predictability of the final EMC results [1][2][6]. Choosing an EMC improved circuit concept as explained in Sect. IV might lead to better results but it does not give a guarantee to achieve noise magnitudes below the EMC limits. A transient simulation of both LDO designs was done with the goal to reproduce the EMC noise levels in first order. The simulation setup was derived from the CISPR 25 standard setup of Fig. 8. A voltage source turned out to be sufficient as battery/supply. A 50Ω resistor was added in parallel to the 1kΩ resistor of the LISN to simulate the 50Ω input of the EMC receiver. The signal $V_{EMC1,2}$ was saved for a few milliseconds. A Fast Fourier Transform (FFT) was performed on $V_{EMC1,2}$. The transient simulation needs to have a high time resolution of less than 5ns (time steps) to obtain a spectrum of up to 100MHz. Most of the waveform tools do not offer a representation in dBµV. The signal has to be multiplied by a factor of 10^6 before performing the FFT to obtain the values in dBµV when a dBV scale is used. The simulation results are shown in Fig. 10. A comparison with Figures 4 and 9 reveals that both simulations reproduce the charge pump noise spectrum and magnitude very well. Especially the most critical fundamental noise peak is predicted with 2-3dBµV accuracy. Fig. 9 also confirms the design considerations from Sect. IV. The difference in the frequency in both simulations compared to the measurements is due to process corner variations of the charge pump clock.

The levels of the noise peaks seem to be always slightly higher than the measured results since the simulation was done pre-layout and does not fully include parasitic capacitance on-chip, of the package and board layout. This indicates that no post-layout simulation is needed, a clear advantage for an early EMC prediction. An additional advantage of the improved charge pump is the compact die size since, due to the constant currents and the increased pumping frequency, the capacitor values can be kept small.

VI. Conclusion

A charge pump design practice for electro-magnetic compatibility has been presented along with a method to predict the EMC performance of a linear voltage regulator already in an early design phase. Charge pump switching currents propagate to supply and ground as well as to the LDO output, leading to noise emissions. Charge pump design recommendations were derived which call for (1) a current-controlled architecture with two-phase operation, (2) a driving capability just at the minimum to meet the load current specification and (3) a pumping frequency within the band around 3-9MHz with less stringent EMC limits. Based on the CISPR EMC standard, the EMC noise level and spectrum can be predicted from a transient simulation. Time steps <5ns and an artificial network termination are needed to achieve a realistic prediction of the EMC noise magnitude in the relevant frequency range. LDOs with two different charge pumps were designed and fabricated in 350nm high-voltage BiCMOS technology. The simulation results are matching well with EMC test results for both circuits with <3dBµV accuracy. Simulation and test show that the proposed current-controlled implementation achieves ~50dBµV less noise than a conventional charge pump with large inrush current peaks.

REFERENCES

[1] Chen, S.; Nehl, T.W.; et al., "Towards EMI prediction of a PM motor drive for automotive applications", *Applied Power Electronics Conference and Exposition, 2003. APEC '03. 18th IEEE Appl. Power El. Conf., APEC'03*, vol. 1, pp. 14-22, 2003.

[2] Perdriau, R.; Ramdani, M.; Levant, J.-L.; Trullemans, A.-M., "EMC evaluation in integrated circuits using VHDL-AMS", *2003 IEEE Int. Symp. on Industrial Electronics*, vol. 1, pp. 193-198, 2004.

[3] Weber, S.; Guttowski, S.; Hoene, E.; John, W.; Reichl, H., "EMI coupling from automotive traction systems", *2003 IEEE Int. Symp. on EMC*, Vol.1, pp. 591-594, 2003.

[4] Christopoulos, C., et al., "Simulation methodologies for electro-magnetic compatibility (EMC) and signal integrity (SI) for system design", *Proc. 7th Packaging Techn. Conf., EPTC 2005*, vol. 2.

[5] Rybak, T.; Steffka, M., "Automotive Electromganetic Compatibility (EMC)", *Kluwer*, 2004.

[6] Wittmann, J.; Wicht, B.; "EMC influence of the charge pump in linear regulators - Design, simulation and measurements*" IEEE Int. Symp. on Circuits and Systems*, pp. 1359 – 1362, 2011.

[7] International Electrotechnical Commission, CISPR 25, 2nded., 2002-08

[8] Lavery, K.; Smith, R., "Impact of Linear Regulator Topology on Integrated Circuit Emissions", *Electromagnetic Compatibility, 2007, EMC 2007, IEEE Int. Symp. on EMC*, 2007.

[9] Berkhout, M.; vanSteenwijk, G.; van Tuijl, A.J.M.; "A low-Ripple Chargepump Circuit for High Voltage Applicaions", *ESSCIRC 1995, Proc. 21th European Solid-State Circuits Conference*, pp. 290-293, 1995.

Juergen Wittmann received his Dipl.-Ing. Degree from the Technical University of Munich, Germany, in 2006. In February 2011, he joined the Robert Bosch Center for Power Electronics at Reutlingen University as a research assistant. He is working towards his Ph.D. degree in the area of power- and micro-electronics.

Bernhard Wicht received the Dipl.-Ing. degree from the Technical University Dresden, Germany in 1996 and the Ph.D. degree from the Technical University Munich, Germany, in 2002.

From 1996 to 1998, he was with MAZ Hamburg GmbH as a designer of analog ASICs. During that time, he was responsible for the design of integrated photo detectors, optical receivers and analog post processing for measurement systems. In 1998, he joined the Institute for Technical Electronics of the Technical University Munich, Germany, as a Research Assistant, where he was working on memory sense amplifiers until 2002 in cooperation with Infineon Technologies. From 2003 until August 2010, he was with the Mixed-Signal Automotive business unit of Texas Instruments, in Freising, Germany. He was leading a development team with focus on various automotive applications such as power management, high-side/low-side drivers, motor control. In September 2010 he became a professor for Integrated Circuits at Reutlingen University, Robert Bosch Center for Power Electronics. His research interests include IC design with focus on power management, gate drivers, motor control, energy efficiency, low-power, ESD, EMC.

He invented ten patents with several more patents pending. Prof. Wicht is member of IEEE and VDE. He also serves as a member of the Technical Program Committee of the European Solid-State Circuits Conference (ESSCIRC).

HOCHSCHULE KONSTANZ
TECHNIK, WIRTSCHAFT UND GESTALTUNG

Entwurf und Aufbau eines zeitkontinuierlichen Bandpass-Delta-Sigma-Modulators vierter Ordnung

Steffen Wälde, Christoph Schick

***Zusammenfassung*—Delta-Sigma-Wandler dienen hauptsächlich in preiswerten Systemen mit moderaten Frequenzen, beispielsweise im Audiobereich, als eine Alternative zu herkömmlichen Analog-Digital-Wandlern. Aufgrund der im Delta-Sigma-Modulator befindlichen digitalen Filter, ist eine Erhöhung des Frequenzbereiches nur schwer zu realisieren. Deshalb wird der Ansatz verfolgt, das digitale Filter im Delta-Sigma-Modulator mittels der Impulsinvarianztransformation analog nachzubilden, um somit leichter höhere Frequenzbereiche erschließen zu können. Dadurch eröffnet sich die Möglichkeit Delta-Sigma-Wandler auch in Gebieten wie der Funkkommunikation einzusetzen. Hier wird ein System vorgestellt, das analoge Signale mit Frequenzen bis 2,5 MHz verarbeitet und einen Bitstream mit einer Bitfrequenz von 10 MHz ausgibt.**

***Schlüsselwörter*—Delta-Sigma, Rauschformung, Impulsinvarianztransformation, Bitstream**

I. Einleitung

Das Delta-Sigma-Verfahren (ΔΣ) ist ein Analog-Digital und Digital-Analog Umsetzungsverfahren, das bereits vor etwa 50 Jahren entwickelt wurde. Solche Umsetzer werden als ΔΣ-Wandler bezeichnet. Zu dieser Zeit war die technische Umsetzung allerdings noch nicht möglich. Erst vor 20 Jahren konnten die ersten ΔΣ-Wandler mittels CMOS Technologie aufgebaut werden. Diese Bausteine finden heutzutage häufig Anwendung in Bereichen mit moderaten Bandbreiten, wie der Audiotechnik in der Unterhaltungselektronik.

Für höhere Frequenzbereiche jedoch sind die internen Komponenten, die in Switched-Capacitor-Technologie realisiert sind, schlecht realisierbar. In dieser Technologie sind die zeitdiskreten Filter des ΔΣ-Modulators aufgebaut. Dennoch werden ΔΣ-Wandler nun zunehmend auch in Bereichen mit hohen Bandbreiten, wie der Funkkommunikation, interessant. Hierbei ist es gewünscht, das empfangene Signal direkt nach der Antenne zu digitalisieren, um den Mischvorgang in die Zwischenfrequenzebene wegzulassen. Gewöhnliche AD-Wandler sind hierzu ebenfalls weniger gut geeignet, da diese in hohen Frequenzbereichen viel Strom benötigen. Deshalb wird der Ansatz verfolgt, das digitale Filter analog nachzubilden. Es wird ein analoges Filter gesucht, dessen Impulsantwort in den Abtastzeitpunkten mit der Impulsantwort des digitalen Filters übereinstimmt.

Steffen Wälde, steffen.waelde@student.kit.edu, Christoph Schick, christoph.schick@htwg-konstanz.de, HTWG Konstanz, Brauneggerstr. 55, 78462 Konstanz.

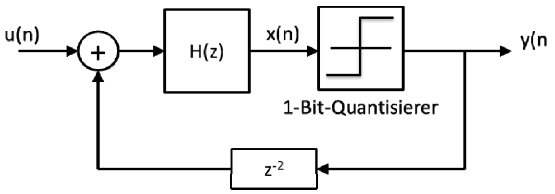


Abbildung 1: Strukturbild eines ΔΣ-Modulators: Als Quantisierer wird ein Ein-Bit-Quantisierer verwendet. Für die negative Phasendrehung sorgt ein Verzögerungsglied in der Rückkopplung, welches die Ausgangsfolge um zwei Folgewerte verzögert. Dieses Blockschaltbild ist Grundlage für den zeitkontinuierlichen ΔΣ-Modulator

II. Schaltungsentwurf

Zunächst wird ein zeitdiskreter Bandpass ΔΣ-Modulator vierter Ordnung vorgegeben, der mittels Impulsinvarianztransformation in einen zeitkontinuierlichen ΔΣ-Modulator überführt werden soll 0. Aus dem Blockschaltbild in Abbildung 1 kann mit vorgegebener Rauschübertragungsfunktion für den ΔΣ-Modulator die Übertragungsfunktion des zeitdiskreten Filters bestimmt werden. Die Übertragungsfunktion des Quantisierungsrauschens soll Bandsperrencharakteristik aufweisen und wird nach folgender Gleichung vorgegeben:

$$N_{TF}(z) = \frac{1}{1 - H(z)z^{-2}} \stackrel{!}{=} \frac{(z^2+1)^2}{z^4}$$

Daraus folgt die Übertragungsfunktion des zeitdiskreten Filters *H(z)*

$$H(z) = \frac{2z^4 + z^2}{(z^2+1)^2}$$

und deren Z-Rücktransformierte, also der Impulsantwort:

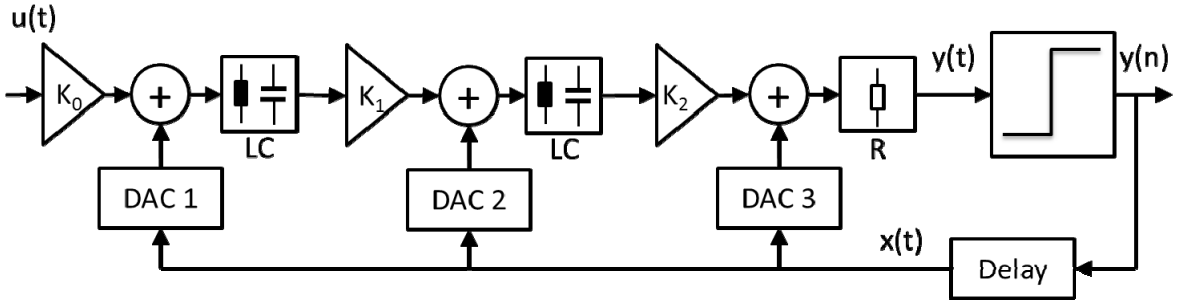


Abbildung 2: Blockschaltbild des zeitkontinuierlichen Delta-Sigma-Modulators: Die Transkonduktanzstufen K_0 bis K_2 sorgen dafür, dass die LC-Resonatoren nicht belastet werden. Über dem Widerstand R fällt die Ausgangsspannung des analogen Filters ab, das Eingangssignal des Ein-Bit-Quantisierers. Das wertdiskrete Signal wird danach zurückgekoppelt, verzögert und über die Digital-Analog-Wandler wieder in die Resonatoren eingespeist.

$$h = \left(2 + \frac{n}{2}\right)\cos\left(\frac{\pi n}{2}\right) \quad \text{mit } n = 0,1,2,\ldots$$

Das benötigte analoge Filter muss so aufgebaut werden, dass in den Abtastzeitpunkten n die analoge Impulsantwort mit der oben angegebenen zeitdiskreten Impulsantwort übereinstimmt.

A. LC-Resonator

Da die Ordnung des zeitdiskreten Filters vier beträgt, bilden zwei identische LC-Resonatoren, deren Resonanzfrequenz bei der Eingangsfrequenz liegt, den Kern des zeitkontinuierlichen Filters. Damit sich die Resonatoren nicht gegenseitig beeinflussen, werden sie durch Transkonduktanzstufen voneinander entkoppelt. Für spätere Berechnungen wird die Übertragungsfunktion der LC-Filter benötigt, welche sich nach folgender Gleichung ergibt:

$$H_{LC}(s) = \frac{1}{C}\frac{s}{s^2 + \omega_0^2}$$

mit Resonanzkreisfrequenz ω_0.

B. Digital-Analog-Wandler

Das rückgekoppelte Signal wird nach dem Verzögerungsglied wieder in die einzelnen Resonatoren eingespeist. Hierzu werden ebenfalls Transkonduktanzstufen verwendet, deren Transkonduktanzen maßgeblich die Impulsantwort des zeitkontinuierlichen Filters bestimmen. In diesem Modell wirken diese Transkonduktanzstufen zusätzlich als Digital-Analog-Wandler, da das rückgekoppelte, digitale Signal in ein analoges Filter eingespeist wird. Deshalb wird für die Impulsantwort eines Digital-Analog-Wandlers ein Rechteckimpuls mit Amplitude I und Zeitdauer T verwendet. Die Übertragungsfunktion berechnet sich dann zu:

$$H_{DAC}(s) = I\frac{1}{s}\left(e^{s\frac{T}{2}} - e^{-s\frac{T}{2}}\right)$$

C. Zeitkontinuierlicher Delta-Sigma-Modulator

Die LC-Resonatoren, Transkonduktanzen und Digital-Analog-Wandler bilden nun zusammen das analoge, zeitkontinuierliche Filter. Für den ΔΣ-Modulator kommen noch ein Quantisierer und ein Verzögerungsglied hinzu, Abb. 2. Über einen zusätzlichen Ausgangswiderstand R kann die Ausgangsspannung des Filters abgegriffen werden. Aus diesem Blockschaltbild kann nun die Übertragungsfunktion des zeitkontinuierlichen Filters hergeleitet werden. Diese ergibt sich nach folgender Formel:

$$H(s) = \frac{Y(s)}{X(s)} = R\frac{1}{s}\left(e^{s\frac{T}{2}} - e^{-s\frac{T}{2}}\right)\cdot$$

$$\left(I_3 + K_2 I_2 \frac{1}{C}\frac{s}{s^2 + \omega_0^2} + K_2 K_1 I_1 \frac{1}{C^2}\frac{s^2}{(s^2 + \omega_0^2)^2}\right)$$

Die Impulsantwort wird nun unter der Voraussetzung:

$$\omega_0 = \frac{2\pi}{4T}$$

für die Fälle $-T/2 \le t < T/2$ und $T/2 \le t$ berechnet. Die Fallunterscheidung ist nicht zwingend notwendig, erleichtert jedoch den Rechenaufwand. Um die Filterkoeffizienten zu bestimmen, müssen die zeitdiskrete und die zeitkontinuierliche Impulsantwort gleichgesetzt und für verschiedene Abtastwerte ausgewertet werden.

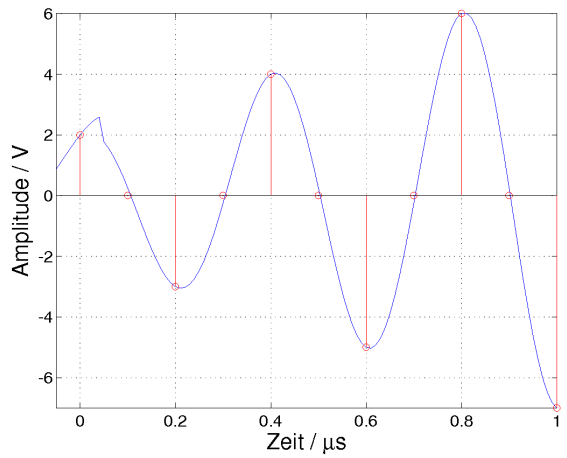


Abbildung 3: Impulsantwort des zeitdiskreten (rot) und des zeitkontinuierlichen (blau) Filters. In den Abtastzeitpunkten stimmen beide Impulsantworten überein. Die zeitkontinuierliche Impulsantwort beginnt aufgrund der Übertragungsfunktion der DACs schon bei -50 ns.

Für $-T/2 \leq t < T/2$ und $n = 0$ ergibt sich folgende Gleichung:

$$\frac{I_1 K_1 K_2 R}{2\omega_0 C^2}\left(t+\frac{T}{2}\right)\sin\left(\omega_0 t+\frac{\pi}{4}\right)+\frac{I_2 K_2 R}{\omega_0 C}\sin\left(\omega_0 t+\frac{\pi}{4}\right)+I_3 R=2$$

Für $t \geq T/2$ und $n > 0$ ergibt sich folgende Gleichung:

$$\frac{I_1 K_1 K_2 R}{2\omega_0 C^2}\left(T\cos\left(\frac{\pi}{4}\right)\sin(\omega_0 t)+2t\sin\left(\frac{\pi}{4}\right)\cos(\omega_0 t)\right)+$$

$$+\frac{2I_2 K_2 R}{\omega_0 C}\left(\sin\left(\frac{\pi}{4}\right)\cos(\omega_0 t)\right)=\left(2+\frac{n}{2}\right)\cos\left(\frac{\pi n}{2}\right)$$

Für die Schaltung werden nun zur Dimensionierung und Bestimmung der Koeffizienten Vorgaben gemacht. Die Bitdauer des Ausgangsbitstreams soll T = 100 ns betragen, die Ausgangsbitfrequenz des Bitstreams beträgt dann 10 MHz. Die Eingangsfrequenz lässt sich über den Zusammenhang zwischen ω0 und T zu 2,5 MHz berechnen. Die Kapazitätswerte der Resonatoren sind jeweils 1 nF und für den Ausgangswiderstand des analogen Filters wird ein 1 kΩ Widerstand verwendet. Aus der Resonanzfrequenz und dem vorgegeben Kapazitätswert erhält man eine Induktivität von 4,05 µH. Die Transkonduktanzwerte K1 und K2 betragen 10 mS, K0 = 1 mS. Somit errechnen sich die Amplituden der Stromimpulse der DACs zu I1 = 1,111 mA, I2 = 2,221 mA und I3 = 0,875 mA.

Die Verzögerungszeit in der Rückkopplung müsste nach Abbildung 1 zwei Zeitschritte zu je 100 ns dauern. Aufgrund der Wahl der Übertragungsfunktion der Digital-Analog-Wandler, muss die Verzögerungszeit nun auf 150 ns verkürzt werden, da diese theoretisch bereits bei -T/2, also -50 ns, beginnt. Die Simulation aus Abbildung 3 bestätigt die berechneten Werte.

III. Schaltungsaufbau

Die Schaltung wurde auf einer Platine mit SMD Bauelementen aufgebaut. Ein vereinfachter Schaltplan ist in Abbildung 4 zu sehen. Als Transkonduktanzstufe wurde der OPA860 von Texas Instruments in SOIC Bauform verwendet. Die Transkonduktanz kann dabei über einen externen Emitterwiderstand und über einen zusätzlichen Steuerpin eingestellt werden. Dieser Chip wird sowohl für die Digital-Analog-Wandler als auch für die Transkonduktanzstufen zwischen den Resonatoren verwendet.

Der getaktete Ein-Bit-Digital-Analog-Wandler besteht aus drei Chips. Zuerst wird das analoge Eingangssignal vom Komparator LT1719 von Linear Technology quantisiert. Die zeitliche Diskretisierung übernimmt das D-Flip-Flop 74V1G79 von STMicroelectronics, das von einem externen Oszillator getaktet wird. Der externe Oszillator LTC6900 von Linear Technology erzeugt dabei ein Rechtecksignal mit einer Taktfrequenz von 10 MHz.

Am Ausgang des D-Flip-Flops liegt der Bitstream an, der zum einen über eine Pufferstufe abgegriffen werden kann, und zum anderen zurückgeführt wird. Das rückgekoppelte Signal wird über zwei integrierte Verzögerungsleitungen (DS1100Z-50+ und DS1100Z-100+) um 150 ns verzögert. Der darauf folgende Operationsverstärker wird zur Pegelanpassung verwendet.

IV. Messungen

Für die Messungen wurde ein Tektronix TDS3034B Vierkanal Oszilloskop verwendet.

A. Impulsantwort

Die Impulsantwort des analogen Filters wurde bei offener Schleife aufgenommen. Angeregt wurde das Filter in dem ein Puls mit einer Dauer von 100 ns auf die Digital-Analog-Wandler gegeben wurde. Das Ergebnis ist in Abbildung 5 zu sehen. Liest man die Frequenz der Sinusschwingung aus dem Diagramm ab, erhält man einen Wert von etwa f_0 = 2,65 MHz. Die gewünschte Resonanzfrequenz liegt jedoch bei f_0 = 2,5 MHz. Die Abweichung kommt daher, dass der Bauteilwert der eingebauten Induktivität nicht genau mit dem berechneten Wert übereinstimmt. Berechnet wurde eine Induktivität von L = 4,05 µH, verwendet wurde jedoch eine Induktivität mit dem Wert L = 3,9 µH mit einer Toleranz von 5 %.

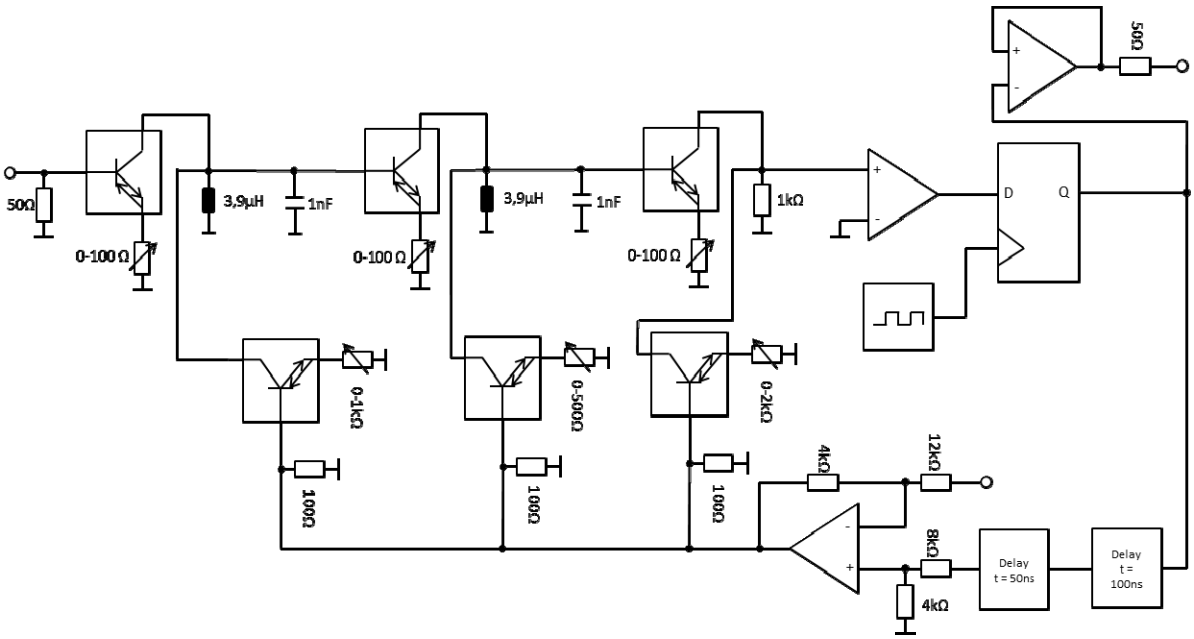


Abbildung 4: Funktionaler Schaltplan des ΔΣ-Modulators: Entstörkondensatoren der ICs und Bauelemente, die nur für den Betrieb der ICs notwendig sind, wurden weggelassen.

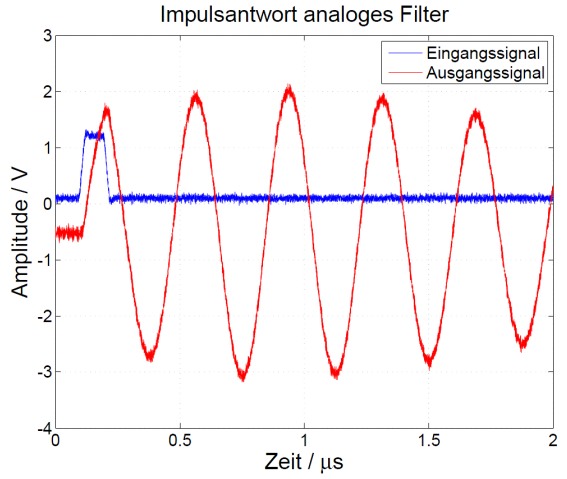


Abbildung 5: Impulsantwort des analogen Filters: Das Eingangssignal (rot) ist ein Impuls mit einer Pulsbreite von 100 ns. Das Ausgangssignal (blau) ist eine abklingende Sinusschwingung mit einer Frequenz von etwa 2,65 MHz

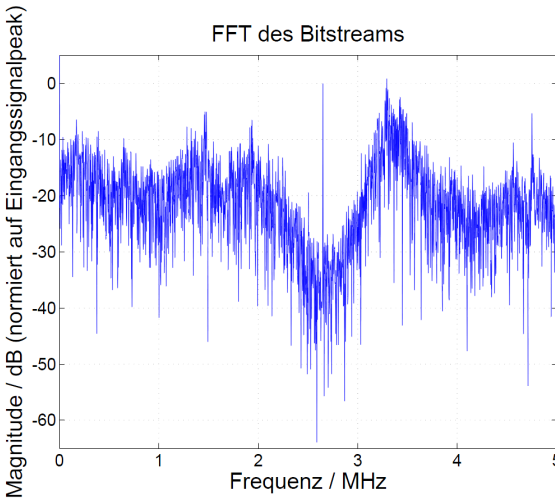


Abbildung 6: Ausschnitt der FFT des Ausgangsbitstreams: Zu sehen ist die Rauschunterdrückung um 2,65 MHz und das Eingangssignal bei 2,65 MHz mit *Uein* = 50 mVpp. Abtastfrequenz: 25 MSamples/s; Anzahl der Abtastwerte: 10000 Punkte

Aus diesem Grund liegt auch später der Rauschtrichter des Spektrums nicht genau bei 2,5 MHz. Zudem beschreibt die berechnete Impulsantwort eine aufklingende Sinusschwingung, die in der aufgebauten Schaltung nicht auftritt. Der Grund hierfür liegt darin, dass die Übertragungsfunktion mit einem idealen, verlustlosen LC-Resonator aufgestellt wurde. Die parasitären Widerstände der Induktivität und der Kapazität, wirken als Dämpfungsglied und führen dazu, dass die Schwingung nach einer e-Funktion abklingt. Bisher ungeklärt ist die Ursache des negativen Offsets des Ausgangssignals.

B. Spektrum

Mit dem Oszilloskop wurde ein Datenstrom mit 10000 Punkten bei einer Abtastrate von 25 MSamples/s aufgenommen und mit Matlab die FFT gebildet. Ein Ausschnitt des Resultates ist in Abbildung 6 zu sehen. Die Frequenz des Eingangssignals betrug bei dieser Messung 2,65 MHz, die Amplitude 25 mV. Man erkennt deutlich die Rauschformung um 2,65 MHz, bei der das Quantisierungsrauschen unterdrückt wird.

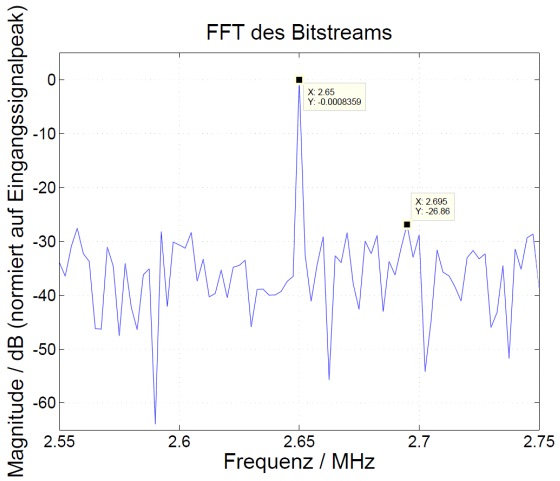


Abbildung 7: FFT des Ausgangsbitstreams: Zu sehen ist die Rauschunterdrückung um 2,65 MHz und das Eingangssignal bei 2,65 MHz mit 50 mVpp. Abtastfrequenz: 25 MSamples/s; Anzahl der Abtastwerte: 10000 Punkte

C. Signal-Rausch-Abstand

Um den Signal-Rausch-Abstand des ΔΣ-Modulators charakterisieren zu können, muss im Vorfeld eine Bandbreite definiert werden. Diese Bandbreite wurde zu 100 kHz gewählt. In Abbildung 7 sieht man den Ausschnitt des diskret fouriertransformierten Ausgangssignals im Bereich von f_0-50 kHz bis f_0+50 kHz. Um nun den Signal-Rausch-Abstand berechnen zu können, wird aus dem Ausschnitt die Differenz des größten Rausch-Werts zum Signal-Wert abgelesen und auf 1 Hz Bandbreite bezogen. Hierzu wird die Bandbreite zwischen den Stützstellen in dB addiert. Die Bandbreite beträgt in diesem Fall B = 2,5 kHz. Somit ergibt sich ein Signal-Rausch-Abstand von 60,9 dB.

V. Fazit

In dieser Arbeit wurde gezeigt, wie aus einem zeitdiskreten ein zeitkontinuierlicher ΔΣ-Modulator hergeleitet wurde. Da es sich um einen ersten Entwurf handelte, lag der Anspruch nicht auf einer sehr hohen Arbeitsfrequenz oder einem großen Signal-Rausch-Abstand, sondern zunächst auf Funktionalität bei einfachem Aufbau.

Der Signal-Rausch-Abstand ist bisher noch relativ gering. Dieser kann durch Optimierung der Verzögerungszeiten und der Resonatorbauelemente oder durch Einfügen einer weiteren Resonatorstufe noch verbessert werden.

Literaturverzeichnis

[1] A. M. Thurston, T. H. Pearce, M. J. Hawksford, “Bandpass Implementation Of The Sigma-Delta Conversion Technique”, *International Conference on Analouge to Digital and Digital to Analouge Conversion*, 1991.

Steffen Wälde erhielt den akademischen Grad des B.Eng. in Elektro- und Informationstechnik im Jahr 2011 von der Hochschule Konstanz.

Christoph Schick erhielt den akademischen Grad des Dipl.-Ing. in Elektro- und Informationstechnik im Jahr 2001 von der Universität Karlsruhe (TH) und den Grad des Dr.-Ing. in Elektrotechnik im Jahr 2007 von der Universität Ulm. Er ist Professor für Hochfrequenztechnik und Elektronik an der Hochschule Konstanz.

MULTI PROJEKT CHIP GRUPPE

Hochschule Aalen
Prof. Dr. Bartel, (07361) 576-4182
manfred.bartel@htw-aalen.de

Hochschule Albstadt-Sigmaringen
Prof. Dr. Rieger, (07431) 579-124
rieger@hs-albsig.de

Hochschule Esslingen
Prof. Dr. Lindermeir, (0711) 397-4221
walter.lindermeir@hs-esslingen.de

Hochschule Furtwangen
Prof. Dr. Rülling, (07723) 920-2503
rue@hs-furtwangen.de

Hochschule Heilbronn
Prof. Dr. Gessler, (07940) 1306-184
gessler@hs-heilbronn.de

Hochschule Karlsruhe
Prof. Dr. Koblitz, (0721) 925-2238
rudolf.koblitz@hs-karlsruhe.de

Hochschule Konstanz
Prof. Dr. Burmberger, (07531) 206-255
gregor.burmberger@htwg-konstanz.de

Hochschule Mannheim
Prof. Dr. Paul, (0621) 292-6351
g.paul@hs-mannheim.de

Hochschule Offenburg
Prof. Dr. Jansen, (0781) 205-267
d.jansen@hs-offenburg.de

Hochschule Pforzheim
Prof. Dr. Kesel, (07231) 28-6567
frank.kesel@hs-pforzheim.de

Hochschule Ravensburg-Weingarten
Prof. Dr. Siggelkow, (0751) 501-9633
siggelkow@hs-weingarten.de

Hochschule Reutlingen
Prof. Dr. Kreutzer, (07121) 271-7059
hans.kreutzer@hochschule-reutlingen.de

Hochschule Ulm
Prof. Dipl.-Phys. Forster, (0731) 50-28338
forster@hs-ulm.de

www.mpc.belwue.de