MPC

MULTI PROJECT CHIP GRUPPE
BADEN-WÜRTTEMBERG

Herausgeber: MPC-Gruppe **Ausgabe:** 64/65 **ISSN:** 1868-9221 **Publiziert:** Mai 2024
Workshop: Esslingen Juni 2022 **Workshop:** Reutlingen Januar 2023

Band 64 Workshop: Esslingen Juni 2022

Band 65 Workshop: Reutlingen Januar 2023

Cooperating Organisation
Solid-State Circuit Society Chapter
IEEE German Section

Inhaltsverzeichnis

Band 64 Workshop: Esslingen Juni 2022

Band 65 Workshop: Reutlingen Januar 2023

Tagungsband zum Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie.

Die Inhalte der einzelnen Beiträge dieses Tagungsbandes liegen in der Verantwortung der jeweiligen Autoren.

Herausgeber:
Lothar Schmidt, MPC-Gruppe, Albert-Einstein-Allee 53, D-89081 Ulm

Mitherausgeber (Peer Reviewer):
Jürgen Giehl, Hochschule Mannheim, Paul-Wittsack-Straße 10, D-68163 Mannheim
Eckhard Hennig, Hochschule Reutlingen, Alteburgstraße 150, 72762 Reutlingen
Frank Kesel, Hochschule Pforzheim, Tiefenbronnerstraße 65, D-75175 Pforzheim
Elke Mackensen, Hochschule Offenburg, Badstraße 24, D-77652 Offenburg
Andreas Siggelkow, Hochschule Ravensburg-Weingarten, Doggenriedstraße, 88250 Weingarten

Alle Rechte vorbehalten

Diesen Workshopband und alle bisherigen Bände finden Sie im Internet unter:
http://www.mpc-gruppe.de/de/workshopbaende.html

HOCHSCHULE ESSLINGEN

Measuring similarity for technical product descriptions with a character-level siamese neural network

Simone Falzone, Tobias Münster, Gabriele Gühring

***Abstract*—As part of the inventory process or when two companies merge and the enterprise resource planning system is taken over, databases with descriptions of technical parts are compared to get an overview of available parts. However, not all entries are created automatically, as the parts are recorded on site by employees and entered manually, with varying part IDs and descriptions. Therefore, it is of interest for a company to use a suitable method that is able to read technical documents and perform an automated comparison between employee information and the inventory data of an Enterprise Resource Planning system. We use a Manhattan Long Short-Term Memory network in order to measure sentence similarity between two entries of an Enterprise Resource Planning systems. The model is evaluated on industry data consisting of technical product descriptions. It outperforms models employing methods of substring analysis, classical machine learning methods as well as deep learning models with pre-trained word-embeddings.**

***Index Terms*—Siamese deep neural networks, Sentence similarity, Character-level, Technical language processing, Technical product descriptions**

I. Introduction

In reconciliation of inventory data all parts must be linked to the physical records. This requires that the digitally recorded items and their characteristics are matched with the items listed in the existing Enterprise Resource Planning (ERP) systems as part of the inventory. During the initial inventory of fixed parts, items are often recorded that are not clearly identified but already have part numbers entered in the ERP system. In some cases even two systems are fused and different part IDs or numbers have to be matched. The challenge in matching is to assign the corresponding data records in the ERP system to the articles found. In addition to tracking inventory data, it is also necessary for a company to keep track of the condition of all parts. The problem, as shown in [1], is that an inventory management system sometimes is fed with new data based on the input of data by maintainers related to equipment inspections, diagnostics and corrective actions. As a result, the data in a Computerized Maintenance Management System (CMMS) may contain unstructured raw text or inconsistent data with, e.g. missing part IDs. Most data in a CMMS can not be used for Natural Language Processing (NLP) diagnostics and analysis, as most out-of-the-box language processing pipelines are designed for non-technical language. In Table I we present examples of technical language description as it can be found in product description databases.

Simone Falzone, Tobias Münster, and Gabriele Gühring Gabriele.Guehring@hs-esslingen.de. Faculty Computer Science and Engineering, University of Applied Sciences Esslingen, 73732 Esslingen, Germany.

Table I
Exemplary descriptions of technical text

No.	Product Description
1	EE80251S1-000U-A99 - Sunon Sleeve Bearing Fan
2	18B11029-TA-1764-1 Tubeaxial Exhaust Fan 1Hp
3	Insulated Ferrule Single Wire 18 Awg

Furthermore, as described in [2], product matching is a crucial aspect for e-commerce portals that combine offers from multiple merchants to allow the user to find the best price for a specific product or efficiently find matching products from a variety of merchants.

In order to analyse technical textual data, such as inventory descriptions, the scientific community has successfully applied Natural Language Processing to its text-based technical data such as an affinity propagation clustering [3], a combination of bidirectional Long Short-Term Memory (LSTM) and conditional random field (CRF) for the task of named entity recognition [4], extracting information from clinical text data using the UMLS MetaMap Transfer (MMTx) application and a negation detection algorithm called NegEx [5].
We present in this paper a deep learning model especially useful for technical data consisting of part numbers, part IDs, units etc., which circumvents common pitfalls by learning important features of the input data, as for example shown in Table I or in Table VI.

II. Related Work

In NLP there are several methods to determine the similarity of words or sentences. A string based approach is presented in [6]. The Basic Local Alignment Search Tool (BLAST) algorithm used there examines two protein or gene sequences for their similarity. This approach can be adapted to regular or technical sentences. It is a dynamic programming approach which

compares sentences at the character level. Equality is defined by the identical sequence of local and global characters in two sentences, with unequal characters being deleted from the sequence.

The approach of [2] uses descriptive statistics for technical product matching from which they combine three metrics in one sum, which additionally can be weighted by the hyperparameters α, β and γ. The Term Frequency-Inverse Document Frequency (TFIDF) is used for describing how often a certain word occurs in a text. The second metric is the Term Partition Relevance (TPR) and describes in which text partition a word occurs most frequently. Finally, the value of the Term Type Partition Relevance (TTPR) is added, which describes the probability of occurrence e.g. of a certain product category in the different text partitions.

In [7] an unsupervised learning algorithm is proposed to solve the problem of matching product titles based on the morphological analysis of the titles of the products. They present an algorithm that operates in two phases. In the first phase combinations of the words of the titles are computed and several statistic measures are recorded. In the second phase we used the gathered statistical information to assign a score to each computed combination. The combination with the highest score is then declared as label of the cluster which contains each product.

The field of NLP has received much public attention due to the successes of language models such as BERT [8], T5 [9] or GPT-3 [10]. One area of NLP is *extractive question-answering*, which can be viewed as a special case of text classification. Given a interrogative sentence and multiple candidate answers, the task is to classify each candidate answer as correct or not [11]. Contrary to [7] our approach is a supervised approach since we need a training data set to train our character embedding. However, it is possible to use our model on different problems and so as in [8] to implement a pretrained character embedding.

Common approaches for computing a semantic embedding of words are GloVe [12] and Word2Vec [13], which create embeddings based on their context and frequency of co-occurence in a text corpus. The idea is to create a vector space in which the position of a word describes how (un-)similar it is to its surrounding words. In our proposed model we do not use a pretrained word embedding, instead the embedding process is done within the training phase.

As for comparing or matching two sentences with each other, given their word or character-level vector representation, siamese neural networks and their deep neural network variants are used [11]. This architecture is utilized in [14], where a similarity metric on variable length character sequences is learned. The model combines multiple character-level bidirectional LSTMs in a siamese architecture. It learns to project variable length sequences into a fixed dimensional embedding space by using information about the similarity between pairs of strings.

In our work we add several convolutional layers to a siamese Manhattan-LSTM (MLSTM) for measuring sentence similarity. Our temporal convolutional neural network is inspired by the architecture of [15], which learns abstract knowledge about words and sentences on the basis of individual characters and generates a word embedding from this. Furthermore, our architecture is inspired by the siamese MLSTM of [16], which reads in two sentences given their word embedding representation using a siamese LSTM and generates a context between the words. From this LSTM representation of the sentences a similarity score is calculated using a similarity layer employing the L1-Norm. We evaluate our model against the established BERT language model with a WordPiece Model (WPM) as presented in [17]. With a WPM, the vocabulary also contains partial words, frequent endings and single numbers and letters. This makes it possible to divide unknown words into smaller known words. Thus, word embeddings can be created for unseen words.

III. Similarity based on WordPiece Model

The comparison of sentences based on their similarity is a well-known problem from the field of Natural Language Processing for which several pre-trained models are already available. One of the breakthroughs in the field of NLP is published in [8] presenting a model called Bidirectional Encoder Representations from Transformers (BERT). We try to establish a baseline with the BERT model and check if, thanks to the WPM used, sentences consisting of non-natural language, i.e. the names and IDs of (electronic) parts in an inventory process, can also be successfully compared.

We use a German BERT-model which is published on Hugging Face[1] and presented in [18], because in our first dataset, presented in section VI-A, the names, IDs or descriptions contain, if at all, mostly German expressions. The model is trained on German Wikipedia entries, the OpenLegalData dump [19] and several German news articles.

Our main goal is comparing two IDs, numbers and names of inventory parts in order to see if they are identical. For the comparison of the two parts, the most important attributes of labeled machine parts are extracted from the data. These include the part number, which uniquely identifies a component, the type of the part which is described in more detail in section VI-A and a short description as well as the manufacturer name. Subsequently, an embedding is created for each attribute, this is done for both parts that are compared with each other. The similarity of two technically described parts is calculated using cosine similarity like in [20] by comparing a word

[1] https://huggingface.co/transformers/pretrained_models.html

Table II
ACCURACY WITH PRE-TRAINED GERMAN BERT-MODEL (DEPENDING ON TRESHOLD T)

T	Accuracy
85	43.912
90	70.220

embedding of each attribute of the first part with the corresponding word embedding of the second part. To obtain the dependency of all attributes, the results of the obtained similarity values are averaged over all attributes.

We have defined two different threshold values for the evaluation, denoted by T. If the calculated similarity of the two parts is greater than or equal to T, they are classified as equal, otherwise as unequal. The $accuracy$ is then calculated for the classified components as in equation (3) section VI-D. The selected values for T and the achieved $accuracy$ is shown in Table II.

The results in Table 2 show that the pre-trained BERT model does not perform well for our use case of the comparison of a technical description of parts. We therefore pursue in the following sections another approach based on generating an intrinsic word embedding with a Convolutional Neural Network (CNN) which leads to higher $accuracy$ on our training and test data set.

IV. DATA AUGMENTATION FOR TEXT BASED DATA SETS

We first use data augmentation for training a more robust model. Data augmentation, commonly used in image processing, encompasses multiple techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built. It acts as a regularizer and helps reduce overfitting when training a machine learning model [21]. Further, we assume that by increasing the amount of text artifacts, such as missing word/letter or swapped words, the robustness and reliability of our model is improved.
To do this, we use aspects of the Easy Data Augmentation (EDA) suite as in [22], which demonstrates performance improvements of NLP models by using augmentation techniques that are loosely subjected to those used in computer vision. The following section describes the EDA suite briefly, when it is applied to real word non-technical sentences.

Each sentence in a database is given the opportunity to be selected for augmentation using one of the following operations, which are chosen at random. Further, synonyms are found in text by employing a dictionary. The synonyms are then processed differently according to the selected operation.

1) **Synonym Replacement (SR)**: Select N_1 words randomly from the sentence that are not stop words. Replace each of these words with a randomly chosen synonym.
2) **Random Insertion (RI)**: Find a random synonym for a non-stop word in the sentence. Insert that synonym into a random position in the sentence. Do this N_2 times.
3) **Random Swap (RS)**: Choose two words in the sentence at random and swap their positions. Do this N_3 times.
4) **Random Deletion (RD)**: Delete each word in the sentence with probability p_{RD}.

Additionally, an individual parameter α_i, for $i = 1 \dots 3$ is calculated for the operations SR, RI and RS, by $\alpha_i = \frac{N_i}{L}$, where L denotes the length of the sentence. Each α_i denotes the amount of words in percent which shall be subjugated to its respective operation.

N_i may vary, depending on the α_i value for its respective operation. This is due to the assumption, that longer sentences can cope easier with manipulation before losing their assigned label. Finally a parameter n_{aug} can be chosen by the user, which indicates how many sentences are to be generated from the original one.
We basically follow the EDA implementation, except that in our algorithm we no longer rely on an English dictionary to identify words. Instead, we split the string by the whitespace character and call each sequence of characters a word. Consequently, we can not and do not use the functionality SR or RI in our setup. As for the parameters α_3, which denotes the amount of characters subject to the RS operation in percent, and p_{RD}, which denotes the probability of an individual character being subject to the RD operation in percent, we select the value of 0.2 for both parameters and set n_{aug} to 9. These parameters are inspired by [22] as reasonable performance gains are to be expected.

V. SIAMESE CNN-MLSTM

In this section we introduce our architecture for measuring whether two inputs of technical product description data classify the same product. The architecture is based on the character-level convolutional neural network presented in [15] for learning important features of the input sentences and the siamese LSTM shown in [16] for finding sequential patterns and calculating a similarity value. Since we are evaluating whether two product descriptions match to the same product we measure the success of our model by calculating an accuracy or an $F\text{-}score$ as in [23] or [24], although we really are interested about a statement of how similar two product descriptions are.

A. Model Architecture

The architecture of the proposed siamese CNN-MLSTM is shown in Fig. 1. The model takes two

Table III
CONVOLUTION LAYERS USED IN OUR ARCHITECTURE. NO LAYER USES STRIDE AND ALL POOLING LAYERS ARE NON-OVERLAPPING.

Layer	Conv1D Filters	Conv1D Kernel Size	MaxPooling1D Size
1	256	7	3
2	256	7	3
3	256	3	N/A

input texts, here named *Sentence 1* and *Sentence 2*, and outputs the probability that both inputs describe the same part. Both sentences are passed into the model in the form of a character embedding matrix. Our siamese CNN-MLSTM consists of several sequential layers: (a) convolution stacked with max-pooling layers, (b) feed forward LSTM, (c) similarity layer and (d) a sigmoid classification layer. Given the input sentences, the convolutional layers learn a specific feature representation of the sentences, which are down-sampled for the maximum presence of a feature, which is then passed to the LSTM layer. The LSTM layer operates on the feature map of the last convolution layer and in turn outputs an internal feature representation from the last LSTM cell, which encodes the sequential patterns. We follow the assumption of [16] that the hidden-state values of the last LSTM cell encode its respective input sentence, all of which is therefore passed into a similarity layer. The similarity layer applies the L1-Norm to both LSTM cells hidden-state values from each input sentence. Its output is finally passed into a sigmoid layer for the classification of whether the input sentences are similar or not. More details about each of the layers are shared in the following subsections.

B. Sentence Encoding

At the beginning, each character of the input sentences of variable length L is converted to lower case and encoded. Here we denote by *sentence* the concatenation of the part description consisting of IDs, names, types, manufacturer, etc. The encoding is done with a fixed alphabet of length m for the language defined here. The sentences are then encoded by a 1-out-of-m encoding (1-hot encoding). The sentence is first divided into an array of individual tokens (characters). Then each token is replaced by its corresponding vector of length m. The result is an array of length L consisting of vectors of length m, which make up the character $m \times L$ embedding matrix mentioned in section V-A. If a sentence exceeds the length defined by $L = 70$, all characters exceeding the limit are ignored. If a sentence is shorter than L characters, it is padded to length L by adding the zero vector. In addition, an UNK (Unknown) token is introduced to avoid equating an out-of-vocabulary error with padding. Therefore our alphabet, minus the UNK token is given by the following characters:

abcdefghijklmnopqrstuvwxyz0123456789-,;.!?:’’’/ \{ }|_@#$%^&*~‘+-=<>()[]

C. Learning Sentence Representation

At the core of the model is a CNN structure, which operates on a character-level input, but outputs a word-vector representation. The idea of the temporal 1D-Convolution is that through its filters a character-n-gram is created. Through stacking convolution operations and clever use of max-pooling, an understanding of word structures/patterns is learned which is specific to the task. The architecture consists of a Conv1D-MaxPooling1D sequence which is described in Table III. The initial weights and biases of the Conv1D elements are normally distributed with the parameters (0.0, 0.05) for mean and standard deviation. The structure of the CNN follows loosely the one presented in [15], but deviating in size of the Conv1D-MaxPooling1D sequence.

D. Sequential Patterns and Sentence Similarity

Subsequently, the resulting feature map of the final convolution layer with the dimensions $d_{in} = 3 \times 256$ is passed to the LSTM layer, which now learns a mapping from dimension d_{in} to dimension $d_{rep} = 64$. With d_{rep} being the dimension of the final vector representation of the sentence, encoded by the last hidden state of the LSTM layer [16]. Finally the LSTM representations of the left and right input sentences are compared by a predefined similarity function $g : \mathbb{R}^{d_{rep}} \times \mathbb{R}^{d_{rep}} \rightarrow \mathbb{R}$ (based on the L1-Norm). Let h_{64}^{left} be the hidden-state representation of the last LSTM cell of the left input and h_{64}^{right} of the right input. With $||x||_1$ denoting the L1-Norm, the similarity is computed as follows

$$g(h_{64}^{left}, h_{64}^{right}) = exp(-||h_{64}^{left} - h_{64}^{right}||_1) \quad (1)$$

The final layer of our architecture is a single sigmoid node, which learns the probability of two inputs being similar or not.

VI. EXPERIMENTAL ANALYSIS

Our model for mapping inventory datasets is trained on two datasets, both describing technical parts or products. First, we use a proprietary dataset provided by a company dealing with machine parts. The second is a publicly available dataset from Kaggle[2] that is similar to the first dataset but consists of descriptions of electrical devices such as smartphones or television. The data is also used in [7] for the matching of product titles in order to compare parts retrieved by a user or marketer by an online search to compare similar products. This is comparable to our use case where we try to compare parts from different data sources,

[2]https://www.kaggle.com/lakritidis/product-clustering-matching-classification

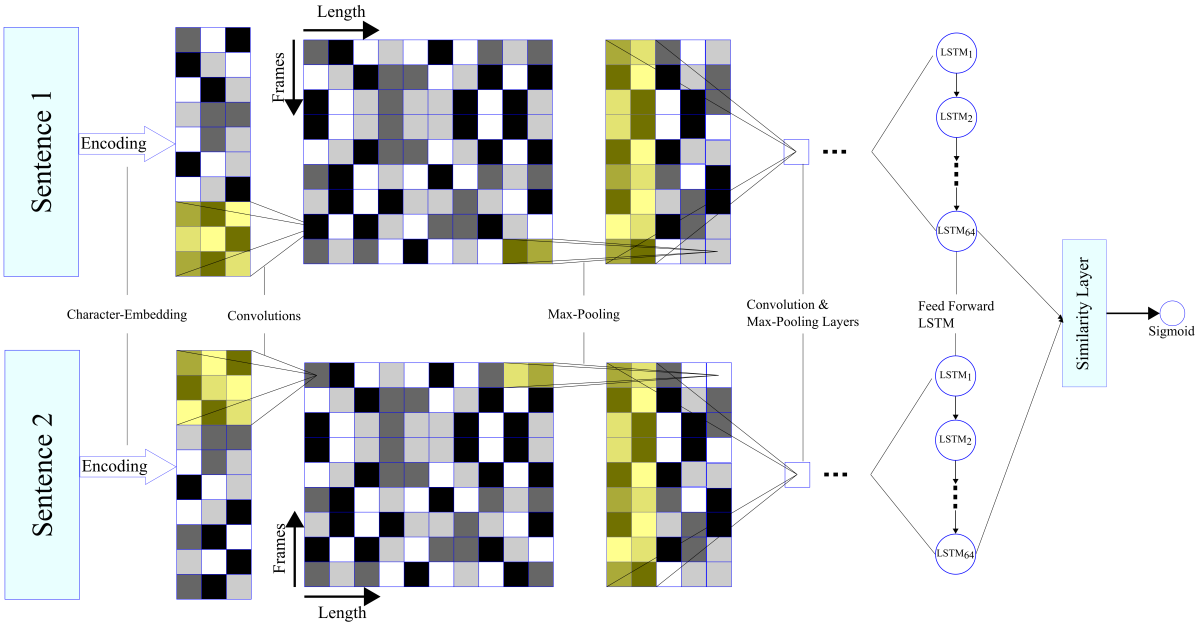


Figure 1. Illustration of our model architecture

Table IV
EXAMPLE OF TWO IDENTICAL PARTS WITH DIFFERENT DESCRIPTION

Sentence 1	**Sentence 2**
MLV12-54-**G32/124** Lichtschranke, Reflexion; RW	208637 **Pepperl + Fuchs** Lichtschranke MLV12-54-**G/32/124** 208637 10 - 30 1 St.

Table V
ATTRIBUTES OF BOTH SENTENCES 1 AND 2

Attribute	**Description**
Part number	Manufacturer part number
Type	Addition to the manufacturer part number to make the part number unique
Description	Contains the title and a description of the part

e.g. from different ERP systems of a company and determine whether both parts are the same or not. Both data sets contain approximately 80.000 records.

A. *Proprietary dataset*

Since our main goal is to match captured parts with existing parts in an ERP system, i. e. as part of an inventory, we have two parts of datasets. First the dataset consisting of the entries in the ERP system, which we consider as the single source of truth. In the following the description of these parts will be referenced as *Sentence 1*. Second there is a list of parts which are labeled differently either because they are manually labeled or because they belong to another system. They contain slight deviations in the description or part number, these parts are referred to as *Sentence 2*. It is for example quite often the case that when entering a component, the part number is entered with spaces instead of hyphens or the product name is prefixed e.g. with the manufacturer brand. Also it possible that in some cases the description is used to add a detailed product description in natural language, which leads to a significant fluctuation in the length of the sentences. An example of two matching parts with a slightly different descriptions is shown in Table IV

For the training dataset we label the Sentence 2 dataset with either 0 or 1, depending on whether the corresponding part matches the entry in the ERP system. Table V shows the attributes of one part and a short description of the attribute.

In order to use the data for our use case it is necessary to transform it into Sentences 1 and 2. Therefore the individual attributes from Table V are concatenated divided by blanks, starting with a part number and ending with a description. The maximum length of a sentence consists of 70 characters, if a sentence is longer it is truncated as described in section V-B. Table VI shows some example records of the resulting dataset.

The data exhibits a strong skewness as there are many more records of non matching parts than there are for matching parts. This means the target $y = 1$ is subrepresented, which could lead to wrong assumptions by the model. Therefore the data is augmented as described in section IV so both values for y are equally distributed.

Table VI
EXAMPLE RECORDS OF TRAINING DATASET FOR PROPRIETARY DATASET

No.	Sentence 1	Sentence 2	Label y
1	4923+14640 A40.891.00 Filter 1020x950x20mm frame	26-4923 Burner Cover - Southbend, SOU1182778	0
2	TIF15.AUXAAC-22596 Touch LED 15 TFT Display	IQ AUTOMATION Flatman TFT Display, Model: FS170A4GSDDH3	0
3	MLV12-54-G32/124 Lichtschranke, Reflexion; RW	208637 Pepperl + Fuchs Lichtschranke MLV12-54-G/32/124 208637 10 - 30 1 St.	1

B. Kaggle Product Dataset

The second dataset we use for training is a publicly provided dataset on Kaggle[3] titled "Product Clustering, Matching & Classification" [7], which we call in the following Kaggle Product Dataset. This dataset is very similar to the proprietary one. It consists of a product title, which already contains all attributes like a part number, the type and a short description of the part. The different parts are always assigned to a certain cluster, which we use here as a single source of truth and corresponds to the entries from the ERP system from section VI-A. Table VII shows a few example records of the Kaggle Product Dataset with the relevant attributes.

The training dataset is constructed by using the cluster label as Sentence 1 and the product title as Sentence 2. Table VII shows sample data records of the Kaggle Product Dataset. The rows of Table VII show an example data record of the Kaggle Product Dataset. Here the product title and the cluster label are describing the same part. Since all parts in the data set are assigned to the correct cluster label, the entries are labelled with $y = 1$. To obtain also non matching records the data is adjusted in a preprocessing step so that a random product is assigned a wrong cluster label and gets the target value $y = 0$, e.g. No. 2 in Table VIII. This gives us a very similar data set as listed in Table VI which compares well with the proprietary data.

C. Model Training

We train the parameters of the model with the objective of maximizing its prediction $accuracy$ given the target labels in a training dataset. For each of our two datasets an independent model is trained and evaluated with a 5-fold cross-validation. The hyperparameters are determined by a simple grid search, choosing the ones that give the best results. We used the Binary Cross-Entropy (BCE) as our loss function. Let $\hat{y}$ be the predicted label and y be the true label then the BCE is calculated as:

$$BCE(\hat{y}, y) = -(y \cdot log(\hat{y}) + (1-y) \cdot log(1-\hat{y})) \quad (2)$$

Finally, the algorithm used for training our model is stochastic gradient descent (SGD) with a minibatch size of 32 and using momentum as in [26], [27]. Our implementation is done using KERAS (TensorFlow 2) [28].

[3]https://www.kaggle.com/lakritidis/product-clustering-matching-classification

Table VII
EXAMPLE RECORDS OF KAGGLE PRODUCT DATASET

No.	Product Title	Cluster Label
1	bosch serie 6 built under freezer in white	Bosch GUD15A50GB Integrated
2	lec cf61lw 60 litre chest freezer white a rated	Lec CF61LW White
3	canon ixus 185 digital camera silver	Canon IXUS 185

Table VIII
EXAMPLE RECORDS OF KAGGLE PRODUCT DATASET

No.	Sentence 1	Sentence 2	Label y
1	bosch serie 6 built under freezer in white	Bosch GUD15A50GB Integrated	1
2	lec cf61lw 60 litre chest freezer white a rated	Bosch GUD15A50GB Integrated	0

D. Experimental Analysis

Our model is evaluated with a 5-fold stratified cross-validation and $accuracy$ is used as metric. Let TP be the true positives and let TN be the true negatives [29], then the $accuracy$ is calculated as.

$$accuracy = \frac{TP + TN}{N}, \quad (3)$$

where N denotes the total count of all records in the dataset [30].

Furthermore, the recall and precision are calculated in order to calculate the $F\text{-}score$. Recall is the fraction of true events and precision is the fraction of detections reported by the model that are correct [30]. Let TN be the false negatives and let FP denote the false positives [29], then the recall and precision are computed as shown in equation (4) and (5).

$$Recall = \frac{TN}{TN + FP} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

Table IX
ACCURACY AND F-SCORE OF OUR SIAMESE CNN-MLSTM

Dataset	Optimizer	Learning-rate	Momentum	Avg. Loss	Avg. $accuracy$	Avg. $F\text{-}score$
Proprietary	SGD	0.1	0.9	0.172	**94.837**	**0.949**
Proprietary	ADAM	0.001	0.9	0.232	92.760	0.929
Kaggle	SGD	0.1	0.9	0.185	93.797	0.937
Kaggle	ADAM	0.001	0.9	0.249	90.944	0.909

Table X
F-SCORES FROM RELATED WORKS

Paper	Method	Dataset	$F\text{-}score$
[2]	Combining several methods from descriptive statistics	Not available	0.5 - 0.67
[25]	Uses tailored approaches for product matching based on a preprocessing of product offers to extract and clean new attributes usable for matching.	Not available	$\sim$ **0.4 - 0.69**
[7]	Morphological analysis of the titles of the products.	Kaggle Product Dataset	0.66

The $F\text{-}score$ is then finally calculated as [30]:

$$F\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6}$$

The results of the 5-fold cross-validation are averaged and can be found in the Table IX.

In each of our model evaluations, we achieve a value above 90% for both Precision and Recall. Several papers deal with similar use cases as we do. As can be seen from Table X our model outperforms all $F\text{-}scores$ achieved by the others. The best comparison is between our results and those of [7], as they are working on the same dataset.

VII. CONCLUSION

In this paper we introduce a new neural network architecture for matching products or product parts with a technical description. Even today, matching product descriptions is a difficult problem in machine learning as also seen in [7] and [25]. The benefit of automated and reliable product matching is not only practically relevant for the matching of manually recorded parts with existing parts in an ERP system it also can be used in online shopping, since it allows the users to compare products from various suppliers. The traditional string similarity metrics do not perform well in product matching as shown in [2]. We also show that pre-trained WordPiece models are not suitable to process technical texts like product descriptions. The reason for this is that the models are trained on a text corpus consisting of natural language texts. In our experiment with a WordPiece model we reached an accuracy of about 70%.

However, it could be successfully shown that a neural network is able to build up its own understanding of word patterns without pre-computed statistics about word distributions. The architecture of the developed model, shown in section V-A, is a new approach for evaluating sentence similarities at a character level. Our model achieved an accuracy of up to 94.8% and an $F\text{-}score$ of up to 93% when working with a simple forward LSTM in the Siamese CNN-MLSTM.

REFERENCES

[1] M. P. Brundage, T. Sexton, M. Hodkiewicz, A. Dima, and S. Lukens, "Technical language processing: Unlocking maintenance knowledge," *Manufacturing Letters*, vol. 27, pp. 42–46, 2021.

[2] A. Thor, "Toward an adaptive string similarity measure for matching product offers," in *INFORMATIK 2010. Service Science – Neue Perspektiven für die Informatik. Band 1*, K.-P. Fähnrich and B. Franczyk, Eds. Bonn: Gesellschaft für Informatik e.V, 2010, pp. 702–710.

[3] X. Chen, H. Xie, F. L. Wang, Z. Liu, J. Xu, and T. Hao, "A bibliometric analysis of natural language processing in medical research," *BMC medical informatics and decision making*, vol. 18, no. Suppl 1, p. 14, 2018.

[4] M. Gridach, "Character-level neural network for biomedical named entity recognition," *Journal of biomedical informatics*, vol. 70, pp. 85–91, 2017.

[5] S. Meystre and P. J. Haug, "Natural language processing to extract medical problems from electronic clinical documents: performance evaluation," *Journal of biomedical informatics*, vol. 39, no. 6, pp. 589–599, 2006.

[6] W. R. Pearson, "An introduction to sequence similarity homology searching," *Current protocols in bioinformatics*, vol. Chapter 3, 2013.

[7] L. Akritidis and P. Bozanis, "Effective Unsupervised Matching of Product Titles with k-Combinations and Permutations," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. Thessaloniki: IEEE, Jul. 2018, pp. 1–10.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners." [Online]. Available: https://arxiv.org/pdf/2005.14165

[11] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–40, 2021.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "Glove: Global vectors for word representation," 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/pdf/1301.3781

[14] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru, "Learning text similarity with siamese recurrent networks," *Proceedings of the 1st Workshop on Representation Learning for NLP*, pp. 148–157, 2016.

[15] X. Zhang and Y. LeCun, "Text understanding from scratch." [Online]. Available: https://arxiv.org/pdf/1502.01710

[16] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 2786–2792.

[17] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Å. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation." [Online]. Available: https://arxiv.org/pdf/1609.08144

[18] B. Chan, S. Schweter, and T. Möller, "German's Next Language Model," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 6788–6796.

[19] M. Ostendorff, T. Blume, and S. Ostendorff, "Towards an Open Platform for Legal Information," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. Virtual Event China: ACM, Aug. 2020, pp. 385–388.

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *arXiv:1908.10084 [cs]*, Aug. 2019.

[21] Connor Shorten and Taghi M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[22] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks." [Online]. Available: https://arxiv.org/pdf/1901.11196

[23] D. Giampiccolo, H. T. Dang, B. Magnini, I. Dagan, E. Caprio, and B. Dolan, "The fourth pascal recognizing textual entailment challenge," in *TAC 2008*. Citeseer, 2008, p. 545.

[24] S. Sultana and I. Biskri, "Identifying similar senteces by using n-grams of characters," in *Recent Trends and Future Technology in Applied Intelligence*. Cham: Springer, 2018, pp. 833–843.

[25] H. Köpcke, A. Thor, S. Thomas, and E. Rahm, "Tailoring entity resolution for matching product offers," in *Proceedings of the 15th International Conference on Extending Database Technology - EDBT '12*. Berlin, Germany: ACM Press, 2012, p. 545.

[26] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[27] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," *International Conference on Machine Learning*, pp. 1139–1147, 2013.

[28] F. Chollet *et al.*, "Keras," 2015.

[29] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. New York ; London: Springer, 2010.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016.

Simone Falzone received his B.Sc. degree in computer science from the Hochschule für Technik Stuttgart in 2020. Since 2020, he is pursuing his Master's degree at Hochschule Esslingen. He is currently working on projects using deep learning for text analytics.

Tobias Münster received his B.Eng. degree in technical computer science from the University Esslingen. In 2021 he obtained his M.Sc. degree in applied computer science from the University Esslingen. Since 2022 he is working in the field of automated driving research.

Gabriele Gühring studied mathematics and physics at the University of Tübingen. After completing her Ph.D. in mathematics she has advised banks and industrial companies throughout Germany in risk controlling, internal models and the valuation of financial derivatives and has supported the implementation of trading systems. She is a professor at Esslingen University of Applied Sciences since 2008 and gives lectures in the subjects of mathematics, statistics and data analytics. Her current publications in the field of machine learning deal with anomaly detection in time series and multimodal models of deep learning.

Hochschule Aalen

Entwurf neuronaler Netze in Matlab und Designflow zur Implementierung auf Intel SoC-FPGAs mit Zugriff auf die Gewichte in dem externen Linux-SDRAM

Berkay Cakir, Heinz-Peter Bürkle

***Zusammenfassung*—Für Maschinelles Lernen mithilfe von Neuronalen Netzen gibt es in Industrie, Medizin oder im Alltag vielfältige Anwendungen. Oft ist auch die Integration in eingebettete Systeme nötig. Auch wenn nicht das Training des Neuronalen Netzes, sondern lediglich die Inferenz im eingebetteten System erfolgt, stellt dies eine Herausforderung bezüglich der Rechenleistung dar. System-on-Chip (SoC) Field Programmable Gate Arrays (FPGAs) gelten als vielversprechende Zielsysteme für neuronale Netze in eingebetteten Systemen, da sie für die spezielle Aufgabenstellung angepasst werden können. Jedoch erweist sich die komplexe Entwurfsmethodik sowie die begrenzte Größe des On-Chip-Speichers oft als problematisch.**

In dieser Arbeit soll ein Designflow vorgestellt werden, welcher eine unkomplizierte Implementierung eines neuronalen Netzes mit externem Linux-SDRAM Zugriff auf einem Intel SoC-FPGA ermöglicht. Dieser Designflow wird beispielhaft anhand dem DE10-Nano als Zielsystem durchgeführt. Das neuronale Netz wird mithilfe von Matlab erstellt und in VHDL-Code umgewandelt. Danach wird dieser Code in ein SoC-FPGA Design mit SDRAM-Transfer eingebunden. Zuletzt erfolgt die Allokation sowie das Beschreiben des Linux-SDRAM-Buffers mit den Gewichten.

***Schlüsselwörter*—SoC, FPGA, neuronale Netze, Matlab, Linux SDRAM**

I. Einleitung

Durch die rasante Entwicklung von Soft- und Hardware hat der Anwendungsbereich von neuronalen Netzen (NN) stark zugenommen. Diese eignen sich insbesondere für die Sprach- sowie Objekterkennung, Optimierungsaufgaben, Klassifikationen und vielen weiteren komplexen Aufgaben[1]. Dabei kommen viele dieser Anwendungsbereiche nicht nur in der Industrie oder Medizintechnik zum Einsatz, sondern mittlerweile auch verstärkt im Alltag.

Neuronale Netze benötigen eine sehr hohe Anzahl von Rechen- sowie Speicherzugriffe. Daher erfolgt deren Implementierung häufig in Graphics Processing Units (GPUs). Eine interessante Alternative hierzu sind FPGAs. Ausschlaggebend dafür sind folgende Gründe:

Berkay Cakir, Berkay.Cakir@studmail.htw-aalen.de, Heinz-Peter Bürkle, Heinz-Peter.Buerkle@hs-aalen.de.Hochschule Aalen, Beethovenstraße 1, 73430 Aalen .

- Hohe Rechenleistung: Neuere Generationen von FPGAs bieten Rechenleistungen, welche in bestimmten Anwendungsfällen GPUs übertreffen [2].
- Energieeffizienz: GPU-basierte Implementierungen haben einen sehr hohen Leistungsverbrauch. High End GPUs wie die *Nvidia Titan rtx*, welche oftmals bei komplexeren neuronalen Netzen verwendet werden, haben eine Verlustleistung von 280 Watt [3]. Im Vergleich dazu hat ein typisches FPGA mit z.B. einen Chip der *Arria 10* Reihe einen Leistungsverbrauch von nur 70 Watt.
- Rekonfiguration: FPGAs sind im Unterschied zu GPUs noch feingranularer und können bei entsprechender Konfiguration ein größeres Maß an Parallelisierung und Pipelining erzielen. Auch kann die Bitbreite genau an die Aufgabenstellung angepasst werden [4].
- Kosten- und Platzersparnis: GPUs benötigen zusätzlich einen Hostcomputer, welcher weitere Kosten verursacht und einen großen Platz einnimmt. Für kleinere eingebettete Systeme sind handelsübliche GPUs in der Regel nicht geeignet. Jedoch bieten Embedded Systems wie z. B. der Jetson Nano eine sinnvolle Alternative.

Die Nutzung von SoC-FPGAs bringt jedoch zwei wesentliche Nachteile mit sich. Der eine Nachteil der FPGAs ist die Speicherproblematik. Denn für viele Anwendungsfälle reicht der On-Chip-Speicher, welcher bei preiswerten Modellen wie z. B. dem DE10-Nano 3 Mbit betragen, nicht aus. Daher muss der Off-Chip-Speicher genutzt werden. Dieser ist bei FPGAs im Gegensatz zu anderen Hardwareplattformen ein limitierender Faktor, wenn eine hohe Performance erzielt werden soll. So kann die geringe Bandbreite des Off-Chip-Speichers zu einem kritischen Bottleneck führen und somit stark die Leistung senken [5]. Daher muss ein wohldurchdachtes Verfahren für den Speichertransfer entwickelt werden. Ein weiterer Nachteil von SoC-FPGAs ist der erforderliche Hardwareentwurf in einer HDL (Hardware Descrition Language). Diese Entwurfsmethodik ist für Softwareentwickler in der Regel nur schwer zugänglich. Die Entwicklung von

neuronalen Netzen sowie die Anbindung von Off-Chip-Speichern mit HDL erfordert große Erfahrungen aufseiten des Entwicklers [6].

Um dem Problem der Entwicklungskomplexität entgegenzuwirken, ist ein aufkommender Trend die High Level Sythesis (HLS). Bei dieser werden FPGAs mit herkömmlichen Programmiersprachen wie OpenCL oder C++ programmiert. Für das Implementieren der Funktionen wird aufgrund der höheren Abstraktionsebene kein spezifisches Wissen über VHDL oder Verilog benötigt [7]. Auch Mathworks bietet für Matlab ein HLS Tool an, um Matlab Code in VHDL oder Verilog Code zu transformieren.

Für den Speichertransfer mit dem Off-Chip-Speicher ermöglicht die Verwendung von Linux Device Treibern und IP-Core Komponenten einen effizienten Entwurf.

In dieser Arbeit soll der gesamte Designflow vorgestellt werden, der zur Realisierung eines neuronalen Netzes mit externem Linux-SDRAM Zugriff auf einem Intel SoC-FPGA führt. Dabei soll das neuronale Netz anhand von Matlab erstellt sowie mithilfe des HLS-Tools in VHDL Code transformiert werden. Daraufhin wird dieser Code zur Synthese der programmierbaren Logik in ein SoC-FPGA Design mit DMA-Transfer auf das Linux-SDRAM eingebunden. Zum Schluss erfolgt die Allokation sowie das Beschreiben des externen Buffers mit den Gewichten mithilfe des Linux Device Treibers *u-dma-buf*.

Im Folgenden werden zuerst die Grundlagen von SoC-FPGAs, neuronalen Netzen, DMA und u-dma-buf erklärt. Im 3. Kapitel geht es um die Erzeugung des neuronalen Netzes als VHDL-Code mithilfe von Matlab. Kapitel 4 erläutert die Einbindung des NNs in ein SoC-FPGA-Design mit SDRAM-Transfer. In Kapitel 5 wird auf die Allokation sowie das Beschreiben der Buffer im Linux-SDRAM eingegangen. Zum Schluss folgt ein kurzes Fazit.

II. Grundlagen

In diesem Kapitel sollen die relevanten Grundlagen dargestellt werden.

A. Neuronale Netze

Aufbau und Funktionsweise von künstlichen neuronalen Netzen leiten sich aus der Vernetzung von Neuronen im Nervensystem von Lebewesen ab. Die NNs bestehen aus mehreren Schichten (Layern). In diesen sind Knotenpunkte enthalten, welche auch Neuronen genannt werden. Die Anzahl dieser kann sich je nach Layer unterscheiden. Die Neuronen der Schichten sind über Gewichtungen miteinander verbunden. Diese bestimmen über die Stärke der Verbindungen zwischen den Neuronen. Während des Trainings werden diese Gewichtungen angepasst. NNs werden prinzipiell in drei Arten von Layern unterteilt: Input Layer, Hidden Layer(s) und Output Layer. Der Input Layer bildet die erste Schicht und enthält die Eingabewerte, welche an die nächste Schicht der Hidden Layer weitergegeben wird. Diese befindet sich zwischen Input und Output Layer. Sind mehrere Hidden Layers enthalten, bezeichnet man das System auch als Deep Learning. In diesen Layern werden die empfangenen Informationen neu gewichtet und zur nächsten Schicht weitergegeben. Der Output Layer ist die letzte Schicht und beinhaltet die Resultate.

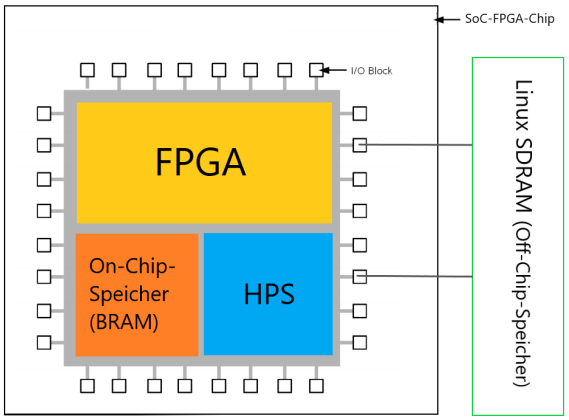


Abbildung 1. Vereinfachter Aufbau SoC-FPGA

B. SoC-FPGAs

SoC-FPGAs vereinen einen Prozessor und eine FPGA-Fabric auf nur einem Chip. Dadurch bieten diese eine höhere Integration, geringeren Leistungsverbrauch, kleinere Boardgröße und eine Kommunikation mit höherer Bandbreite zwischen Prozessor und FPGA [8].

Der Prozessor, auch Hard-Processor-System (HPS) genannt, ist oftmals ein ARM Prozessor mit einem Linux-Betriebssystem. Dieser ist mit dem FPGA über eine AXI Brücke verbunden. Als RAM nutzt das HPS ein SDRAM. Dieses befindet sich nicht auf dem SoC-FPGA-Chip (Off-Chip-Speicher) und hat somit eine höhere Latenz. Jedoch ist SDRAM kostengünstig und wird daher oft bei größeren Datenmengen genutzt.

Gewichte in Neuronalen Netzen lassen sich aufgrund ihres hohen Speicherbedarfs bei größeren Netzen am Besten im SDRAM ablegen. Diese Gewichte lassen sich nun stückweise vom SDRAM auf den On-Chip-Speicher des FPGAs kopieren. Der On-Chip-Speicher wird genutzt, um beim Betrieb des NN die Latenz zu verringern sowie ein Bottleneck zu vermeiden. Als On-Chip-Speicher wird oft BRAM (Block-RAM) verwendet. Abbildung 1 stellt den vereinfachten Aufbau eines SoC-FPGAs mit Speicherzugriff dar.

C. Direct Memory Acess (DMA)

Für das Kopieren der Daten aus dem SDRAM in den On-Chip-Speicher wird DMA genutzt. DMA ist eine Funktion in Computersystemen, die es bestimmten

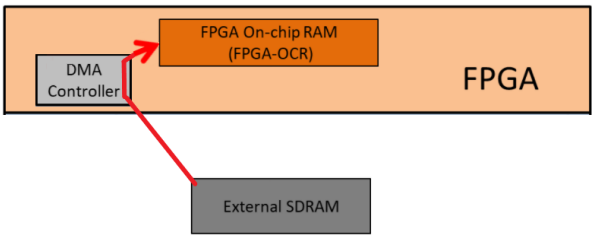


Abbildung 2. DMA-Transfer

Hardware Subsystemen ermöglicht, unabhängig von der CPU auf den Hauptspeicher des Systems (Direktzugriffsspeicher) zuzugreifen. Beim DMA-Transfer werden Daten von einem Adressraum in einen anderen Adressraum kopiert. Ein typisches Beispiel ist das Verschieben eines Blocks aus dem externen Speicher in einen schnelleren Speicherbereich innerhalb eines Systems. Ohne DMA ist die CPU bei einer programmierten Eingabe/Ausgabe in der Regel für die gesamte Dauer des Lese oder Schreibvorgangs voll ausgelastet und steht somit für andere Arbeiten nicht zur Verfügung. Mit DMA leitet die CPU zunächst die Übertragung ein, führt dann andere Operationen durch, während die Übertragung läuft, und erhält schließlich eine Unterbrechung vom Direct Memory Access Controller (DMAC), wenn der Vorgang abgeschlossen ist. Diese Funktion ist immer dann nützlich, wenn die CPU mit der Datenübertragungsrate nicht mithalten kann oder wenn die CPU eine Arbeit ausführen muss [9]. Zusammengefasst ist der Vorteil des DMAs eine schnellere Datenübertragung bei gleichzeitiger Entlastung des Prozessors. Bei SoC-FPGA lassen sich zwei unterschiedliche DMACs nutzen. Einerseits besitzt oftmals das HPS einen DMAC, jedoch kann auch ein DMAC auf Seiten des FPGAs mithilfe einer IP-Core-Komponente angepasst und verwendet werden. In dieser Arbeit wird ein DMAC im FPGA-Design genutzt (wie in Abbildung 2), da dieser eine höhere Datenübertragungsrate bietet als im HPS (ca. Faktor 10).

D. u-dma-buf (User space mappable DMA Buffer)

Die Allokation des Off-Chip-Speichers ist der nächste wichtige Entwicklungsschritt. Dabei muss berücksichtigt werden, dass das Betriebssystem auf dem HPS den Speicher verwaltet. Dieser Speicher wird in zwei verschiedene Bereichen unterteilt: User Space und Kernel Space. Prozesse im User Space haben nur Zugriff auf einen begrenzten Teil des Speichers, während der Kernel Zugriff auf den gesamten Speicher hat. Prozesse im User Space haben zudem auch keinen Zugriff auf den Kernel Space. User Space Prozesse können nur auf einen kleinen Teil des Kernels zugreifen, und zwar über eine Schnittstelle, die der Kernel zur Verfügung stellt, nämlich die Systemaufrufe. Aufgrund des eingeschränkten Zugriffs auf den Speicher im User Space

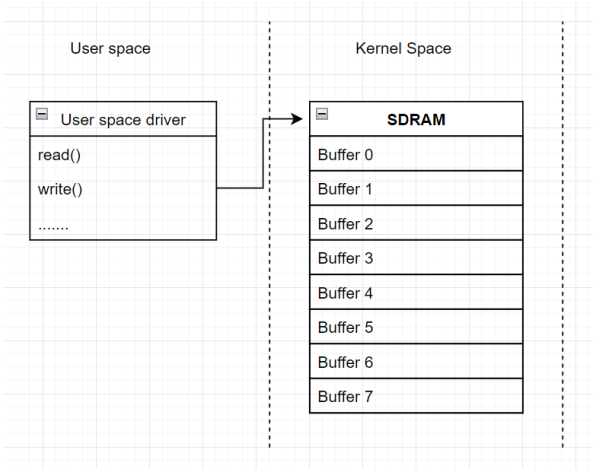


Abbildung 3. Funktionsweise u-dma-buf

und eines nur teilweisen Zugriffs mithilfe von Systemaufrufen erweist sich das Allokieren von Buffern im SDRAM als Äußerst komplex [10]. Jedoch existiert ein Linux Device Treiber mit der Bezeichnung u-dma-buf [11], welcher dies ermöglicht. Mit diesem lassen sich bis zu acht DMA-Buffer im Linux SDRAM (Kernel Space) anlegen, welche vom User space aus verwendet werden können (siehe Abbildung 3).

III. Erzeugung des NNs als VHDL-Code mithilfe von Matlab

In diesem Kapitel soll der Matlab-Designflow zur Entwicklung des VHDL-Codes für ein neuronales Netz beschrieben werden.

A. Erstellung eines NNs in Matlab

Zur Entwicklung neuronaler Netze bietet Matlab die *Deep learning Toolbox* an. Mit dieser lassen sich einfache NNs mit einigen wenigen Layern bis hin zu komplexeren tiefen neuronalen Netzen mit hunderten Schichten erstellen.

Da der Designflow unabhängig von der Größe ist, soll hier ein einfaches Feedforward NN mit zwei Layern genügen. Als Anwendungsfall wird ein Klassifizierungsproblem angenommen werden. Für solche neuronale Netze bietet die Toolbox die *Neural Pattern Recognition* App an.

In der Anwendung müssen zunächst Eingabewerte und deren Ausgabewert für das Training des NN geladen werden. In den folgenden Schritten muss die Verteilung der Daten für das Training, die Validation und das Testen sowie die Anzahl der Neuronen in den Hidden Layer festgelegt werden. Nun kann das neuronale Netz trainiert werden und im Anschluss werden die Resultate ausgegeben. Wenn diese im Rahmen der Anforderungen liegen, kann das NN als Matlab-Code angezeigt und gespeichert werden.

Nachdem das trainierte neuronale Netz erfolgreich als Matlab-Code implementiert wurde, muss dieses

noch entsprechend angepasst und erweitert werden, was für die Konvertierung in Festkommadarstellung und in VHDL erforderlich ist.

B. Anpassung und Erweiterung des NNs

Zuerst soll mit der Erweiterung begonnen werden. Dazu wird eine Top-Layer Funktion erstellt. Diese ist erforderlich, da der HDL-Coder keine Vektoren als direkte Eingabewerte akzeptiert. Deshalb muss eine Funktion implementiert werden, welche die Attribute als Eingabeparameter erhält und diese innerhalb der Funktion in einen Spaltenvektor umwandelt sowie mit dieser die NN-Funktion aufruft. Außerdem soll die Top-Layer-Funktion die Kategorie mit der höchsten Wahrscheinlichkeit ermitteln und diese zurückgeben.

Nachdem die Top-Layer Funktion implementiert wurde, kann der Code innerhalb der NN-Funktion angepasst werden. Zunächst werden unnötige Programmausschnitte entfernt.

Betrachtet man die Unterfunktion *softmax_apply*, ist zu erkennen, dass diese erst prüft, ob die Matrizen auf der GPU berechnet werden. Ist dies der Fall, wird die Funktion *iSoftmaxApplyGPU* aufgerufen. Anderenfalls wird davon ausgegangen, dass die Daten in einer CPU berechnet werden und *iSoftmaxApplyCPU* wird ausgeführt. Die beiden Funktionen verarbeiten die Daten auf unterschiedliche Art weiter, jedoch ergeben sich identische Endergebnisse. Da in diesem Beispiel ein FPGA verwendet wird und somit auch keine Unterscheidung erforderlich ist, ob die Matrizen weder in der CPU oder in der GPU berechnet werden, soll nur eine der beiden möglichen *iSoftmaxApply* Funktionen für die Weiterverarbeitung der Matrizen erhalten bleiben.

Im nächsten Schritt werden unnötige Divisionen entfernt. Diese sollen weitestgehend vermieden werden, da nach der Festkomma Konvertierung die Divisionen mithilfe der Funktion *fi_div* ausgeführt werden. Diese beinhaltet dynamische Matrizen und ist somit stark fehleranfällig bei der VHDL-Umwandlung mit dem HDL-Coder Tool. Oftmals werden die Divisionen für die Berechnung der Softmax gebildet und können entweder vernachlässigt oder im HPS berechnet werden. Falls Divisionen unbedingt benötigt werden, müssen HDL-konforme Workarounds angewendet werden.

Danach werden Exponentialfunktionen, welche ganze Vektoren exponentieren, so umgeschrieben, dass diese jeweils nur noch skalare Größen enthalten. Erst danach sollen die Ergebnisse in Vektoren geschrieben werden. Dieser Schritt wird angewandt, da bei der Festkomma Konvertierung keine Exponentialfunktionen mit Vektoren in Lookup Tables (LUT) umgewandelt werden können.

Zum Schluss werden nicht unterstützte Funktionen ersetzt. Ein häufig vorkommendes Beispiel wären Funktionen des Typs *bsxfun*. Diese führen bestimmte elementweise Operationen an zwei Matrizen durch, was vom Festkomma Konverter nicht unterstützt wird. Daher müssen nun die elementweisen Operationen manuell umgeschrieben werden.

Zusammengefasst sind folgende Schritte notwendig, um das NN-Modell als Matlab-Code für die Konvertierungen anzupassen:

- Erstellung einer Top-Layer Funktion
- Entfernung von nicht benutzten Code-Abschnitten
- Entfernung von nicht essenziellen Divisionen
- Umschreibung von Exponentialfunktionen mit Vektoren zu skalaren Größen
- Umschreibung von nicht unterstützten Funktionen wie z. B. bsxfun

Diese Schritte können je nach NN-Modell variieren oder abweichen. Dennoch bietet die Liste einen groben Überblick worauf zu achten ist, wenn ein NN konvertiert werden soll.

C. Erzeugung von VHDL-Code

Die Erzeugung des VHDL-Codes findet mithilfe des Matlab Tools *HDL-Coder* statt. Es wird zuerst eine Festkomma Konvertierung vorgenommen. Die Festkommadarstellung hat den großen Vorteil, dass mit dieser eine höhere Leistung erzielt werden kann. Der Nachteil jedoch ist, dass der Wertebereich begrenzt dargestellt wird. Außerdem wird die Festkommadarstellung für die Implementierung auf dem FPGA benötigt.

Im *Fixed Point Converter* Tool müssen zuerst die Exponentialfunktionen zu LUTs umgewandelt werden. Im nächsten Schritt wird der Matlab-Code analysiert und die vorgeschlagene Wort- sowie Bruchlänge kann angepasst werden. Danach kann der Prozess gestartet werden. Im Anschluss wird eine Übersicht mit der Abweichung des festkommakonvertierten Modells zum vorherigen Modell angezeigt.

Eine Abweichung zwischen den Modellen kann auftreten. Diese mögliche Differenz ist vor allem den Exponentialfunktionen, welche bei der Festkommakonvertierung zu LUTs umgewandelt werden, zuzurechnen. Bei einem NN Modell mit vielen Neuronen könnte die Modellgenauigkeit zwar besser ausfallen, jedoch würde dieses im Gegenzug mehr Exponentialfunktionen besitzen, welche zu LUTs umgewandelt werden müssten und die Abweichungen im Festkomma System wären daher umso größer. Daher ist es wichtig, diese Differenz zu überprüfen sowie den Tradeoff Effekt zwischen höherer Modellgenauigkeit und Abweichung in Festkommadarstellung bei der Parametrisierung der Neuronen zu berücksichtigen.

Nun kann mit der Erzeugung des VHDL-Codes begonnen werden. Dazu muss im nächsten Schritt das Zielgerät sowie das Synthesetool ausgewählt werden. Danach kann die Konvertierung gestartet werden und die benötigten VHDL-Dateien werden generiert.

IV. Einbindung des NNs in ein SoC-FPGA Design mit SDRAM-Transfer

In diesem Kapitel soll der Workflow für ein FPGA Design mit SDRAM Kommunikation, DMAC und On-Chip-Buffer beschrieben werden. Dazu wird zuerst auf den Aufbau im *Quartus Platform Designer*, gefolgt von einem möglichen Zustandsautomaten als Ablaufsteuerung zur Berechnung des NNs eingegangen.

A. SoC-FPGA Architektur mit SDRAM-Transfer

Für die Einbindung der Datenübertragung wird der Quartus *Platform Designer* genutzt. Mit diesem lässt sich leicht ein FPGA-Design für die SDRAM Nutzung erstellen. Das Design besteht aus folgenden fünf Komponenten:

- HPS (hps_0)
- On-Chip-Buffer (onchip_memory2_0)
- Clock mit 50 MHz (clk_0)
- PLL (pll_0)
- DMAC (dma_0)

Im ersten Schritt müssen alle diese Komponenten in das Design eingefügt werden. Danach werden diese wie folgt konfiguriert. Abbildung 4 zeigt die Verbindungen der Komponenten.

Zuerst wird die Clock Source auf 50 MHz gesetzt und mit der PLL verbunden. Bei dieser wird die Ausgangsfrequenz auf 100 MHz eingestellt. Der Vorteil der höheren Taktfrequenz ist, dass Lese und Schreibzugriffe, welche zwei Takte benötigen, innerhalb eines Taktes der Grundfrequenz von 50 MHz abgeschlossen sind.

Anschließend wird das *FPGA-to-HPS SDRAM* Interface im HPS aktiviert. Dazu wird ein Port mit dem Namen *f2h_sdram0* mit der Typenbezeichnung *Avalon-MM Bidirectional* hinzugefügt. Dieses Interface wird genutzt, um auf die Daten im SDRAM zuzugreifen. Nun müssen noch die verschiedenen Clocks im HPS mit der PLL verbunden werden. Beim DMAC müssen keine besonderen Konfigurationen gesetzt werden. Als Clock wird der PLL verwendet. Der *control_port_slave* ist für die Steuerung der DMAC verantwortlich. Dieser Port wird exportiert, um die Steuerung im FPGA durchzuführen. Der *read_master* liest die Daten aus und wird daher mit dem SDRAM Interface verbunden. Zum Schluss wird der *write_master* mit dem On-Chip-Buffer verknüpft, da in diesem die Daten zwischengespeichert sowie von diesem aus verarbeitet werden sollen. Zudem wird für den On-Chip-Speicher noch die Option *Dual-Port-Access* aktiviert, da zwei Slaves benötigt werden: Ein Port ist für das Speichern der Daten vom DMAC und ein weiterer exportierter Port ist für das Auslesen des Buffers in der programmierbaren FPGA-Logik vorgesehen.

Abschließend wird die Clock des On-Chip-Speichers mit der PLL verknüpft. Optional kann der Slave-port für das Beschreiben des Buffers mit der AXI Brücke verbunden werden, um diesen im Anwendungsprogramm auszulesen oder zu debuggen.

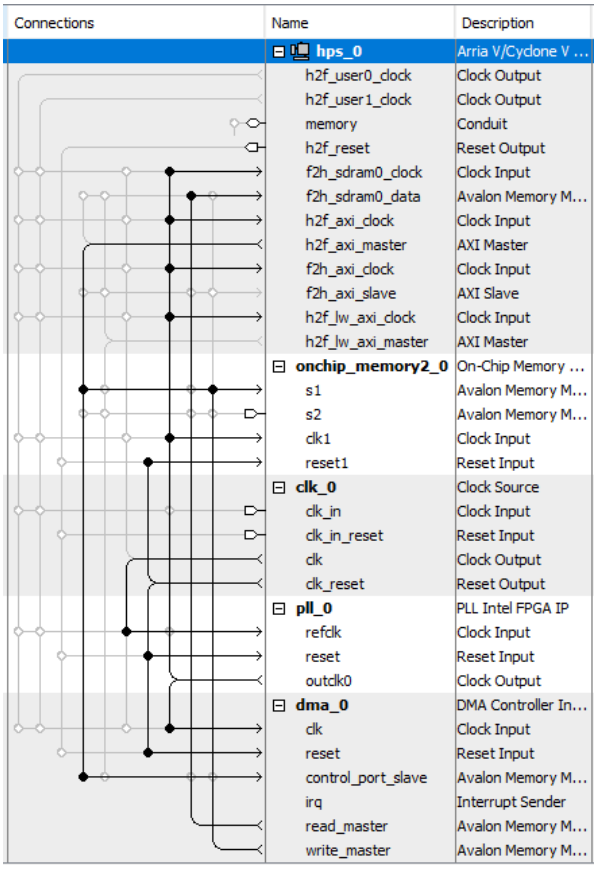


Abbildung 4. SoC-FPGA-Design im Platform Designer

B. Implementierung des Zustandsautomaten

Im Folgenden soll ein möglicher Zustandsautomat vorgestellt werden, welcher die Kommunikation mit dem SDRAM für den FPGA steuert und den Algorithmus des NNs ausführt. Der Zustandsautomat ist von vielen verschiedenen Faktoren abhängig wie z. B. der Größe des NNs. Bei tiefen neuronalen Netzen müssen die einzelnen Teilnetze partiell berechnet werden, da aufgrund der Größe nicht die gesamten Gewichte und LUTs im On-Chip-Buffer gespeichert werden können. Ein beispielhafter Automat kann wie in Abbildung 5 aussehen.

Im Rahmen der vorliegenden Arbeit lässt sich das NN mit nur zwei Layern innerhalb eines Zustandes als Ganzes ausführen und wurde einfachheitshalber auch mit einem solchen Zustandsautomaten implementiert.

Bei der Implementierung der Automaten muss unbedingt die richtige Taktfrequenz für die Zustände ermittelt und ggf. angepasst werden.

V. Reservierung von Buffern im Linux SDRAM mit u-dma-buf

In diesem Abschnitt wird die Implementierung des Linux Device Treibers *u-dma-buf* beschrieben. Dazu

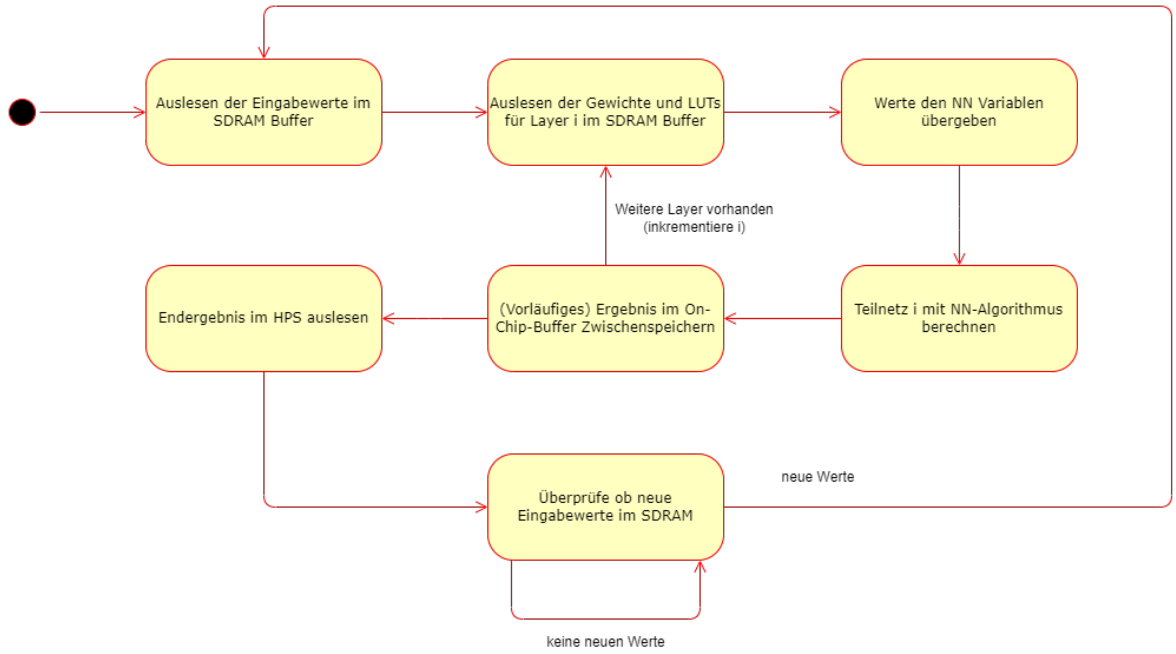


Abbildung 5. FPGA Zustandsautomat für größere neuronale Netze mit partieller Berechnung von Teilnetzen auf dem FPGA

wird zuerst dargestellt, wie durch diesen Treiber der Off-Chip-Speicherplatz allokiert wird. Danach folgt eine Erläuterung, wie dieser Buffer mithilfe eines Anwendungsprogramms im HPS beschrieben wird.

A. Allokation der Buffer mithilfe des Linux Device Treibers

Für die Implementierung des *u-dma-buf* Treibers wird für das Linux Betriebssystem im HPS bestimmte Kernelversionen vorausgesetzt (3.18, 4.4, 4.8, 4.12, 4.14, 4.19, 5.4). Falls keines dieser Versionen vorhanden ist, muss zuerst eine passende Kernelversion kompiliert werden.

Nachdem das Kernelimage erfolgreich kompiliert wurde, kann nun der Device Treiber für *u-dma-buf* erstellt werden.

Mit diesem Treiber kann im HPS des SoC-FPGAs nun SDRAM-Speicher allokiert werden. Dazu gibt es es drei verschiedene Möglichkeiten.

Die erste Methode ist die Konfiguration mithilfe des Device Tree Files. Mit dem Device Tree können Treiber bereits beim Booten des HPS geladen werden. Dabei können im Device Tree die Anzahl, die Größe und sogar die Adressen der Buffer und vieles mehr definiert werden.

Die naheliegende Möglichkeit ist es, den Speicher mit dem *u-dma-buf* Manager zu allokieren. Dabei kann der Treiber mit dem Anwendungsprogramm geladen und dann der Speicher für das NN erstellt werden.

Die nachfolgende Methode ist die schnellste und einfachste Möglichkeit: Der Device Treiber erstellt mithilfe des *insmod* Befehls in der Linux-Konsole die Buffer.

B. HPS Software für das Befüllen der Buffer

Zuletzt soll die Software für den HPS erstellt werden. Diese soll den Buffer im SDRAM beschreiben sowie den On-Chip-Buffer des FPGAs auslesen.

Zuerst werden die virtuellen Adressen für den Zugriff des HPS auf die programmierbare Logik des FPGA mithilfe von mmap generiert. Es wird ein Adressraum von 1 GB abgebildet, der den Bereich von 0xC0000000 bis 0xFFFFFFFF abdeckt. Durch dieses Mapping können nun die einzelnen virtuellen Adressen der betreffenden Module auf dem FPGA berechnet werden.

Im nächsten Schritt müssen im FPGA Register *FPGA2DRAMC* bestimmte Konfigurationsbits auf 1 gesetzt werden. Dies ist notwendig, damit sich die FPGA-to-SDRAM Ports nicht zurücksetzen. Standardmäßig befinden sich diese in einem Reset-Zustand und können dann nicht genutzt werden.

Nun wird in der Software auf den Linux Device Treiber zugegriffen und folgende Aktionen ausgeführt:

Der CPU-Cache wird zuerst deaktiviert. Dies ist notwendig, da die Daten zwischen dem Prozessor und dem DMAC nicht kohärent sind.

Danach dürfen die allokierten SDRAM-Buffer beschrieben werden. Dies geschieht durch eine einfache *read* Anweisung.

Nachdem der externe Linux-SDRAM allokiert und beschrieben wurde, kann das FPGA-Design mit dem NN in Betrieb genommen werden.

VI. Fazit

Im Rahmen dieser Arbeit wurde erfolgreich ein Designflow geschaffen und implementiert, mit wel-

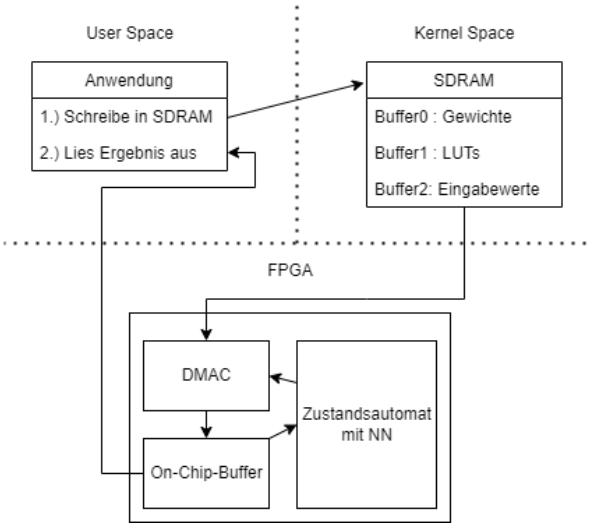


Abbildung 6. Aufbau und Funktion des Gesamtsystems

chem ein in Matlab erstelltes NN auf einem Intel SoC-FPGA mit Zugriff auf Linux-SDRAM entwickelt werden kann.

Dabei wurde aufgezeigt, wie ein solches NN-Modell mithilfe von Matlab erstellt und angepasst werden kann. Ein FPGA Design, welches mit einem DMA-Transfer den Inhalt des Linux-SDRAMs in den On-Chip-Speicher des FPGAs kopiert, wurde erstellt. Durch den Linux Device Treiber u-dma-buf können bis zu acht Buffer im SDRAM angelegt und beschrieben werden. Abbildung 6 zeigt den Aufbau und die Funktionsweise des Gesamtsystems.

Dieser Designflow ermöglicht eine unkomplizierte Implementierung eines NNs. Mit der Nutzung eines Off-Chip-Buffers wird der limitierende Faktor des geringen On-Chip-Buffers für das Speichern der Gewichte von tiefen neuronalen Netzen hinfällig.

Der Workflow und das System wurden auf dem DE10-Nano mit einem Feedforward NN mit zwei Layern getestet, lässt sich jedoch ohne Probleme auf weitere *Cyclone 5* SoC-FPGA Boards und größere neuronale Netze übertragen.

LITERATURVERZEICHNIS

[1] Michaela Tiedemann: *KI, künstliche neuronale Netze, Machine Learning, Deep Learning: Wir bringen Licht in die Begriffe rund um das Thema „Künstliche Intelligenz“* URL: https://tinyurl.com/2whbwt2v - Letzter Zugriff 05.01.2021.

[2] Eriko Nurvitadhi u. a. : *"Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?“ In: Proceedings of the 2017 ACM/SIGDA Interna- tional Symposium on Field-Programmable Gate Arrays. FPGA '17. Monterey, California, USA: Association for Computing Machinery, 2017, S. 5–14. ISBN: 9781450343541.* URL: https://doi.org/10.1145/3020078.3021740 - Letzter Zugriff 05.01.2021.

[3] Nvidia *Titan rtx Specifications* URL: https://www.nvidia.com/content/dam/en-zz/Solutions/titan/documents/titan-rtx-for-creators-us-nvidia-1011126-r6-web.pdf - Letzter Zugriff 04.01.2021

[4] Intel *FPGA vs. GPU for Deep Learning* URL: https://tinyurl.com/28atztht - Letzter Zugriff 04.01.2021

[5] Xuechao Wei, Yun Liang und Jason Cong: *„Overcoming Data Transfer Bottlen- ecks in FPGA-based DNN Accelerators via Layer Conscious Memory Manage- ment“. In: 2019 56th ACM/IEEE Design Automation Conference (DAC). 2019, S. 1–6.* - Letzter Zugriff 22.02.2022

[6] Atze van der Ploeg: *Why use an FPGA instead of a CPU or GPU? 2018.* URL: https://blog.esciencecenter.nl/why-use-an-fpga-instead-ofa-cpu-or-gpu-b234cd4f309c - Letzter Zugriff 04.01.2021

[7] Margit Kuther Andreas Widder: *OpenCL – neuer Ansatz für die Programmierung von SoC-FPGAs. 2016.* URL:https://www.embedded-software-engineering.de/opencl-neuer-ansatz-fuer-die-programmierung-von-soc-fpgas/ - Letzter Zugriff 04.01.2021

[8] Intel. *What is an SoC FPGA?* URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ab/ab1_soc_fpga.pdf - Letzter Zugriff 05.01.2021

[9] FPGAKey: *DMA Dircet Memory Access.* URL: https://www.fpgakey.com/wiki/details/308 Letzter Zugriff 20.02.2022.

[10] User Space vs. Kernel Space. URL: https://unix.stackexchange.com/questions/87625/what-is-difference-between-user-space-andkernel-space - Letzter Zugriff 20.02.2022.

[11] Github: *u-dma-buf.* URL: https://github.com/ikwzm/udmabuf - Letzter Zugriff 20.02.2022.

Berkay Cakir Berkay Cakir erhielt den akademischen Grad des Bachelor of Engineering in Elektrotechnik im Jahr 2021 von der Hochschule Aalen. Dort studiert er seit 2021 den Master Machine Learning and Data Analytics. Aktuell arbeitet er an seiner Masterarbeit im Bereich Predictive Maintenance.

Heinz-Peter Bürkle Heinz-Peter Bürkle bekam den akademischen Grad Dipl.-Ing. in Elektrotechnik im Jahr 1991 von der Universität Stuttgart und den Grad Dr.-Ing. 1997 ebenfalls von der Universität Stuttgart. Nach einigen Jahren in Forschung und Entwicklung bei Alcatel ist er seit 2003 Professor für Schaltungstechnik und rechnergestützten Schaltkreisentwurf an der Hochschule Aalen. Als Prorektor Digitalisierung treibt er zudem diverse Projekte im Feld der Künstlichen Intelligenz voran.

Hochschule Reutlingen
Reutlingen University

Design of an ASIC for Sensorless Control of a Switched Reluctance Motor

Erik John, Till Moldenhauer, Lukas Riedel, Zong Xern Sim, Eckhard Hennig

***Abstract*—Switched Reluctance Motor (SRM) drives have gained interests over the last few years due to being low-cost and robust as well as not utilizing permanent magnets. However, the control schemes necessary for SRMs typically require expensive hardware and complex software. A recent novel control scheme overcomes these problems by injecting a current additionally in another phase and evaluating the corresponding current controller's PWM frequency by digital logic gates. The work in this paper adapts and expands this idea by implementing this method in an application-specific integrated circuit (ASIC) in a 350 nm technology for further reducing system cost as well as increasing portability. The ASIC integrates three main functionalities necessary for motor control: phase sequencing, current control and position detection. Full performance is achieved in simulations but also in the laboratory on real hardware at least up to 5800 revolutions per minute on a DC link voltage of 325 V. In conclusion the developed ASIC is the first of its kind and allows system designers to implement a SRM motor control faster and thereby spreading the application range.**

***Index Terms*—Switched Reluctance Motor, SRM, Motor Control, ASIC, Power Electronics**

I. Introduction

Switched reluctance motors are characterized by a cost-effective and robust construction due to not utilizing permanent magnets. Moreover, this is a benefit under the light of sustainability because the materials for permanent magnets are often mined noxiously. Therefore, SRMs offer a lot of potential in different product areas such as kitchen equipment or even in automotive applications. However, the control system needed for the application of SRMs typically consists of expensive hardware and/or complex software thereby reducing the system cost-effectiveness. One possibility to reduce costs is to control the SRM without position detection sensors, which is the area of interest for this paper. According to [5], three general sensorless control schemes can be currently identified:

1) *Observer based methods*
   In [7] the authors showcase a state observer derived from the linear state equation. The work in [9] exploits the nonlinear relationship between flux linkage, current and rotor position while implementing a sliding mode position observer. Observer based methods are typically characterized by the necessary non-negligible computational power [5].
2) *Incremental inductance based methods*
   These methods rely on the fact that the incremental inductance is proportional to the phase current rise and fall times. From that the rotor position can be derived [5]. Recent research in this area is presented in [4], where a digital signal processor is used for motor control.
3) *Direct inductance based methods*
   The third method of controlling a SRM makes use of observing the motor inductance. The inductance and, consequently, the rotor position is determined by evaluating the current signal.
   The work described in [1] deals with the injection of a DC pulse into an unused phase to determine the phase inductance. By processing an inductance vector, which is set up using a Clark transformation, in a PLL, the rotor position and speed can be calculated.
   The method in [3] is based on the measurement and comparison of two consecutive switch-on times. The approach is characterised by its simplicity and resistance to disturbances in dynamic operation.

The novel control algorithm proposed in [11] utilizes simple digital logic components to estimate the rotor position based on the switching frequency of the hysteresis phase current controller. Besides the suitability of this algorithm to be implemented in an ASIC, there is also no other integrated circuit (IC) currently available in the market for SRM control. In [6] and [2], SRM control ASICs are described, but in contrast to ASIC presented in this paper, they do not feature internal rotor position detection and rely on external position sensors. Consequently, the proposed SRM control ASIC could contribute to reducing system costs and development resources for this motor and, thus, to extending the application range.

Erik John, erik.john@student.reutlingen-university.de, Till Moldenhauer, till.moldenhauer@student.reutlingen-university.de, Lukas Riedel, lukas.riedel@student.reutlingen-university.de, Zong Xern Sim, zong_xern.sim@student.reutlingen-university.de, Eckhard Hennig, eckhard.hennig@reutlingen-university.de. Reutlingen University, Alteburgstraße 150, 72762 Reutlingen.

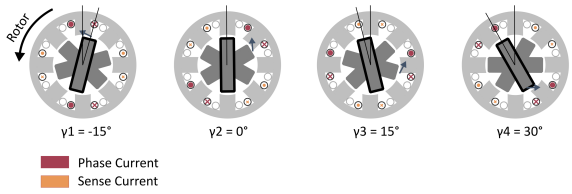


Figure 1. SRM working principle

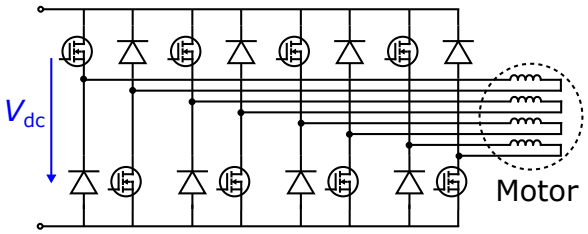


Figure 2. Asymmetric half-bridge

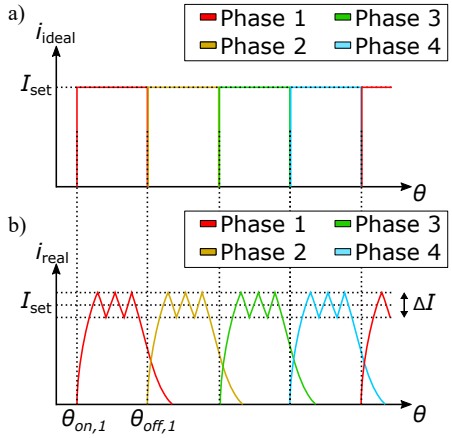


Figure 3. Current signals

II. Theoretical Background

A. General operation of a SRM

The torque generation in SRMs can be derived from the tendency of a magnetic circuit to assume its state of minimal reluctance. In a SRM, the rotor consists of discrete poles and, thereby, variable reluctance in the active magnetic circuit when spinning. The stator is composed of distinct coil halves in opposition to each other, which form the active magnetic circuit when current is flowing through that coil. As shown in Figure 1, the rotor pole most closely to the active stator coil (Red) aligns itself. A continuous rotation can be achieved by switching between the coils periodically, where the correct timing is of crucial importance [8].

B. The power electronic circuit

As already indicated in Section II-A, the torque is generated by driving current through the stator coils.

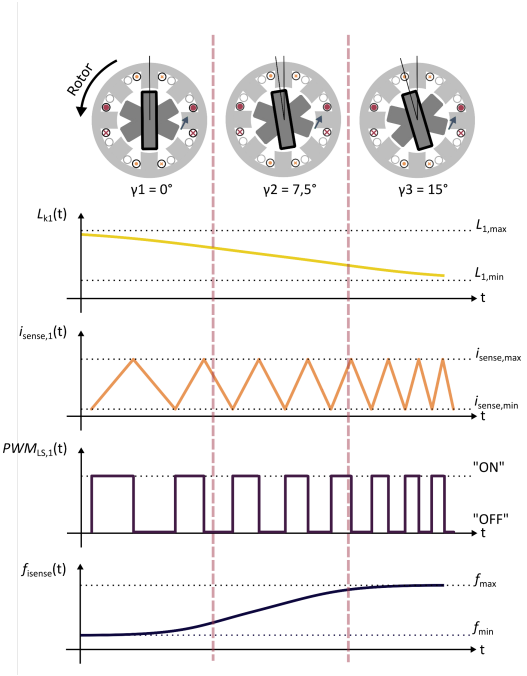


Figure 4. Connection between switching frequency and rotor position

The typical circuit to fulfill this task is called the asymmetric half-bridge, shown in Figure 2. With this circuit configuration, it is possible to drive nearly ideal block currents through the coil. Due to the inductance of the motor, though, the real current signals feature slopes and a hysteresis around the setpoint (see Figure 3). Fast demagnetization can be achieved by turning off both switches on one of the coils so that the negative DC link voltage occurs across that coil. The current is typically controlled via PWM signals with variable frequency [8].

C. The novel sensorless method for SRM control

The variable inductance when rotating leads to rising and falling PWM frequencies, from which the time instants for switching phases can be derived.

The authors of [11] recognized that using the torque generating coil for position sensing also does not lead to robust results, especially in higher speed modes. Therefore, they propose to use one of the inactive phases as the rotor position sensing coil by driving a small current through it. This can be seen in Figure 4. The reluctance of the torque generating coil (red) decreases and thereby its inductance increases while the rotor aligns itself. Contrary, the reluctance of the sensing phase (orange) increases and its inductance decreases because the rotor moves away. This movement can be seen in the rising PWM frequency of the sensing current controller. Utilizing the PWM signal and not the current itself facilitates the evaluation by simple digital hardware. The pairing of torque generating and sensing phase is constant and chosen so that the

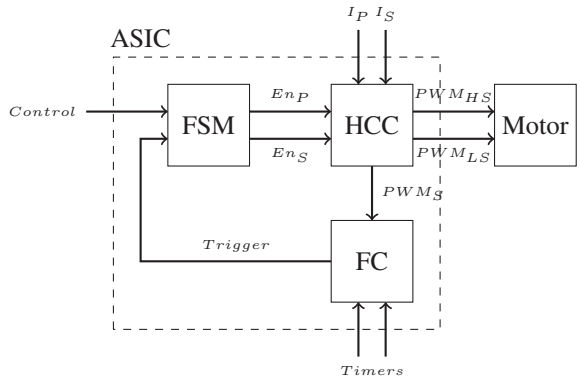


Figure 5. Block diagram

sensing phase is least aligned for the best position estimation. For the utilized SRM the phases 1 and 3 as well as 2 and 4 are paired alternating between power and sensing mode.

III. System Design

The method to develop the ASIC can be described as a top-down approach. Firstly, the partition of the motor control function into smaller sub-functions is done on system level resulting in a block diagram. Subsequently, the sub-functions are detailed, developed and simulated according to an analog and digital design flow. The design phase concludes with the verification of the parasitic extracted layout in simulations.

A. Project conditions

Besides the main requirement for the ASIC to control a SRM with eight stator teeth, the ASIC shall be able to work as a stand-alone system not needing any form of microprocessor or similiar for the basic operation. Moreover, the operation parameters such as the turning direction, the current setpoint values and frequency trigger points shall be configurable. Each main block shall be separately testable.

B. Block diagram

According to the technique to control the SRM in [11], there are three main functionalities, which must be handled by the ASIC:

- Determining the phases to be energized and their sequence.
- Controlling currents and generating the pulse signals for the asymmetrical half bridge.
- Detecting frequency thresholds that lead to phase current switching.

Therefore, as shown in the block diagram, three main modules are introduced (see Figure 5).

a) Finite state machine (FSM): Each of the four power states of the motor can be represented as part of a state transition diagram. In each state, the power phase and sensing phase are in a fixed local relationship to one another (see Chapter II-C). An internal trigger signal causes the state machine to change its state. External control signals can be used to start, stop and reset. Depending on the state of the machine, corresponding enable signals for the power phase and the sensing phase to be measured are passed on to the following block.

b) Hysteresis current control (HCC): Depending on an external current measurement in the powered motor phases, a current control is carried out in the chip. The current is controlled with a fixed hysteresis of $100\,\mathrm{mA}$ around the reference currents set for the power phase and sensing phase. For this purpose, the transistors of the asymmetrical half bridges are controlled with the PWM signals generated in the hysteresis controller. Within the ASIC, the PWM signals of the current controllers for the sensing phase are synthesized and transmitted to the third main module.

c) Frequency comparator (FC): To determine the right time for the phase change, the transmitted PWM signal of the sensing phase is monitored in this block. For this purpose, the time between two signal edges is measured and compared with a set timer value. If its frequency is greater than the upper hysteresis threshold, the internal trigger signal is set to low. This causes the state machine to switch to the next motor state. In order to use the ASIC for different motor inductances, voltages and speeds, the values of the timers can be changed individually.

The output signals of each of the three main components are connected to chip pads, so they can be monitored or replaced by external signals.

IV. Circuit Design and Simulation

The circuit design of the ASIC is divided into an analog and a digital workflow. While the analog design is only used for the hysteresis current controller, the digital workflow is applied for the other two main components, as well as some interconnection elements.

A. Analog design workflow

Each of the four half bridges has to be driven by one high side and one low side PWM signal to achieve the desired current flow with a hysteresis of $100\,\mathrm{mA}$. A Schmitt Trigger consisting of a comparator with two resistors serves this purpose. The hierarchical circuit is then implemented using XFAB's IP components. A functional schematic of the HCC of one phase is shown in Figure 6. The setup consists of two inverting Schmitt triggers per phase, which are connected via various logic gates to generate the required PWM signals of the high and low side.

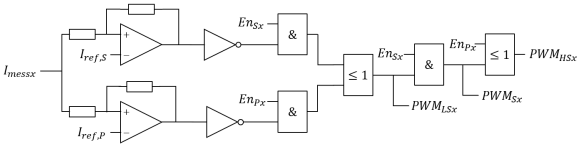


Figure 6. Schematic HCC for one phase

B. Digital design workflow

The digital circuitry is controlled by a 1 MHz clock providing marginal angle error with respect to the SRM's expected maximum speed. The digital circuits are described in the hardware description language VHDL and synthesized as a digital circuit using the Cadence Genus tool.
As shown in Chapter III, the logic to control the phase activation and sequencing can be described as a Moore machine. The FC is implemented using timers to measure the time between two rising PWM edges. Furthermore smaller logic circuits are responsible for blending the four PWM signals for each halfbridge into one glitch free signal as the FC's input. All distinct blocks are connected with each other on a top level description forming one monolithic digital design.

C. System simulation

In order to be able to carry out the simulations to prove the correct functions of all components in the ASIC, a motor model must be built that corresponds to the SRM's behaviour (Figure 7). This is done by qualitatively describing the SRM's angle-dependent inductance in VHDL-AMS by a hyperbolic function.
The equation for the angle dependent inductance is:

$$L = L_{amp} \cdot (1.0 + \tanh(a \cdot \cos(\alpha \cdot \Theta \cdot b))) + L_{min}$$

with
$\alpha = konst. = \frac{T \cdot \pi}{180°}$
$L_{amp} = konst. = 0.5 \cdot (L_{max} - L_{min})$
a: Slope steepness
b: Phase shift
Θ: Rotation angle

Figure 8 shows the resulting inductance. The modelled inductance curves deviate from the real ones at the top, but for the intended simulations only the inductance range, the slope steepness as well as the relative value to each other (crossing points) are relevant. Hence, the qualitative model meets these requirements and is appropriate for the intended application.

Consequently, all the resulting function groups are combined with the motor model into one system simulation model. The simulation of the start-up and operating behaviour provides the results shown in Figure 9.

Initially, no phase is energized. When the system is enabled, phases 1 and 3 are used to align the motor and

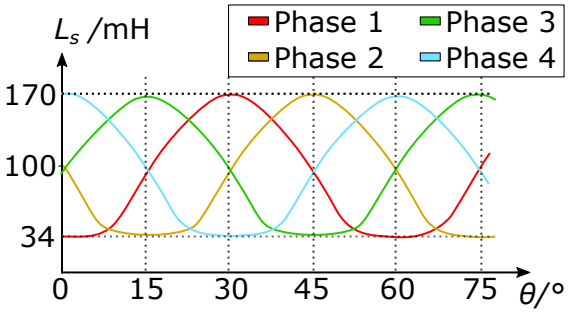


Figure 7. Real inductance curve [10]

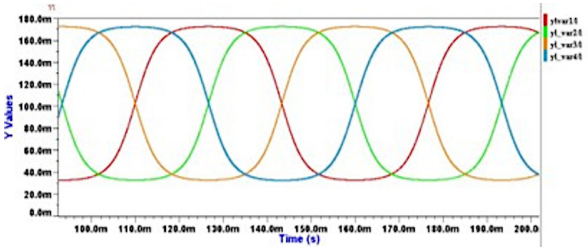


Figure 8. Modelled inductance curve

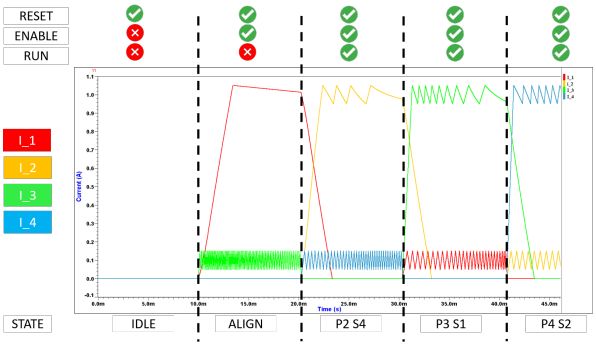


Figure 9. System simulation result

put it in a defined state. The motor can start from this state. Subsequently, the current control is successfully carried out in all phases and the frequency thresholds are detected in the sensing phase, so that a phase change occurs.

V. Layout Design and Verification

The ASIC has been designed and fabricated in XFAB's XH035 350 nm CMOS process with three metal layers. The chip area is 2158 µm x 1880 µm and the Cadence Virtuoso view is displayed in Figure 10.

The core design constitutes the center part of the ASIC whereas the ESD frame forms the outer boundary. Between those two instances buffer capacitors have been placed. The core design's bottom part is formed by the digital circuitry. The digital layout is implemented using the Cadence Innovus toolchain. On the left hand side one can find the ADCs used to set the frequency comparator limits as IP blocks. On the

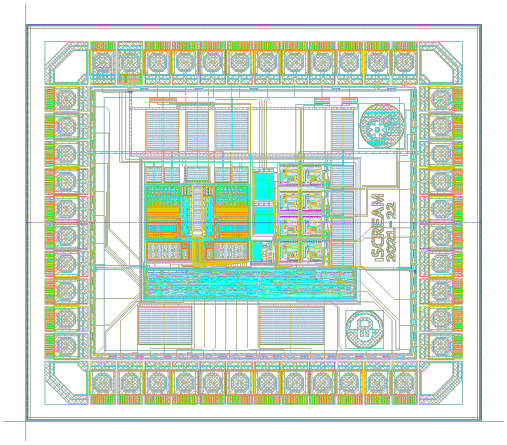


Figure 10. ASIC physical design

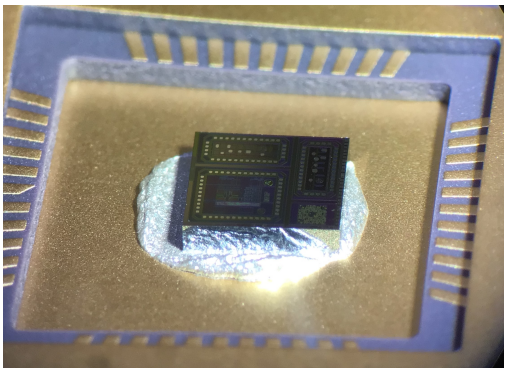


Figure 11. Manufactured die with the SRM control ASIC design in the bottom left corner

right hand side there are the current controller circuits with comparator IP blocks.

During the layout phase, specific attention has been laid upon achieving a compact floorplan, short trace lengths as well as matching similar components to gain robustness against production tolerances and other external influences. The ASIC design presented in this paper has been placed on one die with other chip designs. The result after manufacturing and before bonding can be seen in Figure 11.

VI. Commissioning

The main functionality of the ASIC is to control a system which can drive a SRM. For this purpose, the ASIC is placed in a suitable power electronics system consisting of multiple printed circuit boards (PCBs) displayed in Figure 12. Via gate driver ICs, the ASIC's output signals control four asymmetric half bridges connected to the SRM's stator coils. The current is sensed by hall effect sensors delivering a corresponding signal to the ASIC. Low interference on the analog signals as well as secure transmission of the control signals is crucial for the functionality.

Figure 13 shows the results of the ASIC's successful operation. The purple signal represents the PWM_C

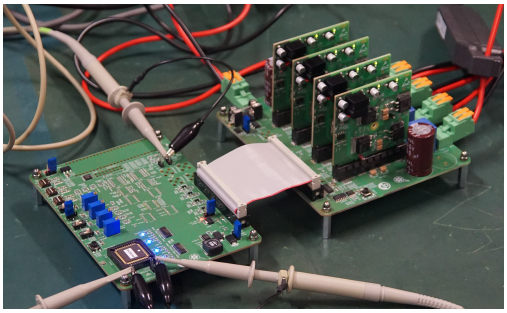


Figure 12. ASIC in the power electronic system

signal resulting from the merger of the four PWM signals (one for each half bridge) into one signal which is fed to the FC. The pink signal ($Trig$) is the output of the FC, indicating the moment in time to switch phases. With every rising edge of the brown signal (RPM), one complete cycle through the state machine is finished. This documents the correct cycling. The four current signals on the top of Figure 13 display the two possible alternating states of the motor phases. When used in power mode, the current does not reach its maximum due to the low DC bus voltage and, therefore, no hysteresis can be seen. When used in sense mode, the hysteresis is clearly visible, proving the correct functionality of the hysteresis current controller inside the ASIC. Moreover the pairing of phase 1 and 3 as well as 2 and 4 becomes apparent. The increased noise in phases 2 and 4 is due to the measurement by current clamps with different resolutions.

VII. Conclusion

This paper shows the possibility to put a sensorless control algorithm into an ASIC, which can be used with a variety of 8/6 SRMs because of the configurability via analog input signals. The ability to control a SRM supplied with mains voltage makes the ASIC suitable for real world applications. Due to facilitating the application of this motor type by integrating several functions necessary for motor control designers do not have to think about those anymore. The described functions comprise position detection and current control but not the speed control. Therefore further work on this chip could mean implementing speed control and initial position detection. Furthermore the ability to work not only with 8/6 SRMs but other motor setups could be included.

VIII. Acknowledgement

The work displayed in this paper has been generously financed by the MPC group and the authors would like to express their gratitude here.

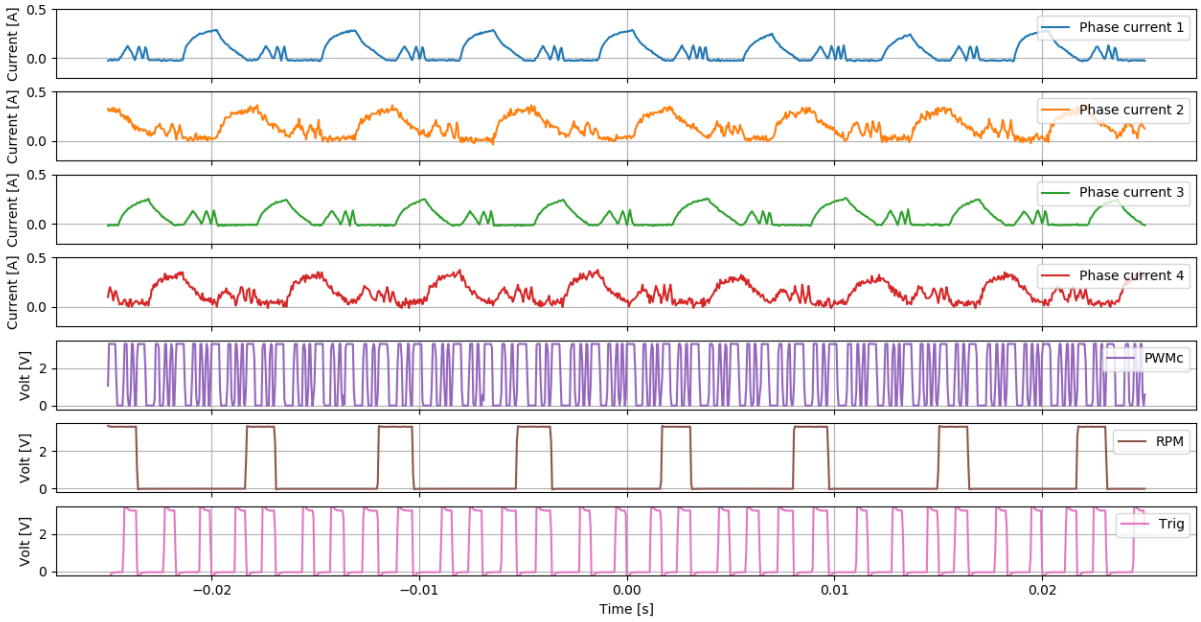


Figure 13. ASIC performance proven in the laboratory

References

[1] Alecksey Anuchin u. a. „Self-sensing Control of a Switched Reluctance Drive Using Voltage Pulse Injection“. In: *2018 X International Conference on Electrical Power Drive Systems (ICEPDS)*. IEEE, Dez. 2018. DOI: 10.1109/ICEPDS.2018.8571654.

[2] Hai-Jin Chen, Sheng-Li Lu und Long-Xing Shi. „Development and validation of a general-purpose ASIC chip for the control of switched reluctance machines“. In: *Energy Conversion and Management* 50.3 (März 2009), S. 592–599. DOI: 10.1016/j.enconman.2008.10.013. URL: https://doi.org/10.1016/j.enconman.2008.10.013.

[3] Jaehyuck Kim, Hyong Yeol Yang und R. Krishnan. „Parameter Insensitive Sensorless Control of Single Controllable-Switch-Based Switched Reluctance Motor Drive“. In: *8th International Conference on Power Electronics - ECCE Asia*. IEEE, Dez. 2011. DOI: 10.1109/ICPE.2011.5944561.

[4] Jongwan Kim und Jih-Sheng Lai. „Quad Sampling Incremental Inductance Measurement Through Current Loop for Switched Reluctance Motor“. In: *IEEE Transactions on Instrumentation and Measurement* 69.7 (2020), S. 4251–4257. DOI: 10.1109/TIM.2019.2949319.

[5] R. Krishnan. *Switched Reluctance Motor Drives - Modeling, Simulation, Analysis, Design, and Applications*. Boca Raton, Fla: CRC Press, 2017. ISBN: 978-1-351-83595-4.

[6] T.J.E. Miller, C. Cossar und D. Anderson. „A new control IC for switched reluctance motor drives“. In: *1990 Fourth International Conference on Power Electronics and Variable-Speed Drives (Conf. Publ. No. 324)*. 1990, S. 331–335.

[7] Yoshihiro Nakazawa und Serina Matsunaga. „Position Sensorless Control of Switched Reluctance Motor Using State Observer“. In: *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*. 2019, S. 1–4. DOI: 10.1109/ICEMS.2019.8921668.

[8] Dierk Schröder und Joachim Böcker, Hrsg. *Elektrische Antriebe – Regelung von Antriebssystemen*. Springer Berlin Heidelberg, 2021. DOI: 10.1007/978-3-662-62700-6. URL: https://doi.org/10.1007/978-3-662-62700-6.

[9] Xiaodong Sun u. a. „Position Sensorless Control of Switched Reluctance Motor Drives Based on a New Sliding Mode Observer Using Fourier Flux Linkage Model“. In: *IEEE Transactions on Energy Conversion* 37.2 (2022), S. 978–988. DOI: 10.1109/TEC.2021.3125494.

[10] Annika Walz-Lange. „Entwicklung einer leistungselektronischen Baugruppe zur Ansteuerung einer geschalteten Reluktanzmaschine“. Bachelorthesis. Reutlingen University, 2020.

[11] Annika Walz-Lange und Gernot Schullerus. „Sensorless Control of a Switched Reluctance Machine Based on Switching Frequency Evaluation“. In: *2020 XI International Conference on Electrical Power Drive Systems (ICEPDS)*. IEEE, Okt. 2020. DOI: 10.1109/icepds47235.2020.9249339.

Erik John received his Bachelor degree in Electrical Engineering in 2021 from DHBW Stuttgart. Currently he pursues his Master degree in Power- and Microelectronics at Reutlingen University. Recent activities include among other things the ASIC development for switched reluctance motors.

Till Moldenhauer received his Bachelor degree in Electrical Engineering in 2019 from DHBW Stuttgart. Currently he pursues his Master degree in Power- and Microelectronics at Reutlingen University. Recent activities include among other things the ASIC development for switched reluctance motors.

Lukas Riedel received his Bachelor degree in Electrical Engineering in 2019 from DHBW Ravensburg. Currently he pursues his Master degree in Power- and Microelectronics at Reutlingen University. Recent activities include among other things the ASIC development for switched reluctance motors.

Zong Xern Sim received his Bachelor degree in Electrical Engineering in 2020 from Mannheim University of Applied Sciences. Currently he pursues his Master degree in Power- and Microelectronics at Reutlingen University. Recent activities include among other things the ASIC development for switched reluctance motors.

Eckhard Hennig received the Dipl.-Ing. degree in electrical engineering from the Technical University of Braunschweig, Germany, in 1994 and the Dr.-Ing. degree from the University of Kaiserlautern, Germany, in 2000. He is a Professor of digital and integrated circuits with Reutlingen University, Germany. His research interests include low-power CMOS circuit design for smart-sensor applications and electronic design automation.

Hochschule Reutlingen
Reutlingen University

Hardware-Plattform für die FPGA-Emulation von Analog-/Mixed-Signal-Systemen

Philipp Czerwenka, Tobias Wolfer, Eckhard Hennig

***Zusammenfassung*—Zur Forschung und Entwicklung von Modulatoren für die Leistungselektronik wird das Systemkonzept einer Emulator-in-the-Loop-Plattform vorgestellt. In diesem Paper werden das Design und die Verifikation der dafür benötigten Hardware präsentiert. Zur Emulation der Analog-/Mixed-Signal-Modulatoren wird ein hochperformantes FPGA verwendet. Die modulare Hardware besteht daneben aus analogen und digitalen Schnittstellen, der Beispielapplikation, sowie Spannungsversorgungsmodulen. Das analoge Frontend weist eine Signalgüte von $65{,}48\,\mathrm{dB}$ auf, einen störungsfreien Dynamikbereich von $-73{,}17\,\mathrm{dB}$ und einen Rauschboden von ca. $100\,\mathrm{dB}$. Die maximale Emulationfrequenz liegt bei $11{,}37\,\mathrm{MHz}$. Der Eingangsspannungsbereich liegt bei bis zu $220\,\mathrm{V}$. Die Implementierung eines beispielhaften Delta-Sigma Modulators wurde nachgewiesen.**

***Schlüsselwörter*—Analog Emulation, Modulator, Datenakquise, FPGA, Mixed-Signal, Hardware-Design**

I. Einleitung

Modulatoren sind elektronische Systeme, die ein Nutzsignal in einem Trägersignal durch Veränderung dessen Eigenschaften kodieren. Je nach Modulationsverfahren sind Modulatoren analoge, digitale oder gemischt analog-digitale Systeme mit entsprechenden Ein- und Ausgangssignalen. Der Einsatz von Modulatoren in der Kommunikationstechnik zur Übertragung und zum Multiplexing von Signalen ist bekannt [1]. In der Leistungselektronik werden Modulatoren in Klasse D Verstärkern eingesetzt [2].

Die Erforschung von Modulatoren ist komplex, da diese Mixed-Signal-Systeme große Frequenzbereiche umfassen und oft nichtlineare Komponenten aufweisen. Die Analyse mittels System- oder Schaltungssimulation liefert oftmals aussagekräftige Ergebnisse, kann jedoch nicht in Echtzeit erfolgen und erfordert große Rechnerressourcen [3]. Eine Implementierung als diskrete Schaltung auf Leiterplatte oder integrierter Schaltung in Silizium ist die bisherige Alternative [4], die zu realitätsnahem Verifikation befähigt, jedoch mit hohem finanziellem und entwicklungstechnischen Aufwand verbunden ist.

Philipp Czerwenka, philipp.czerwenka@reutlingen-university.de, Tobias Wolfer, tobias.wolfer@reutlingen-university.de, Eckhard Hennig, eckard.hennig@reutlingen-university.de. Hochschule Reutlingen, Lehr- und Forschungszentrum Electronics & Drives, Oferdinger Straße 50, 72768 Rommelsbach.

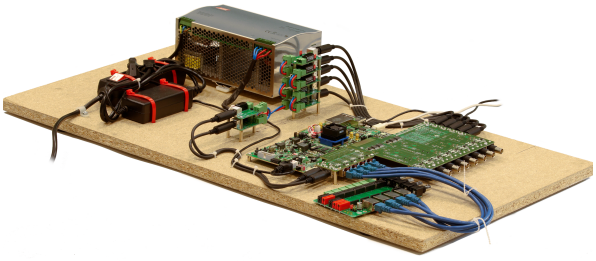


Abbildung 1. Laboraufbau der Hardware des umgesetzten Emulationssystems

Ein neuer Ansatz bietet die Emulation der Schaltungsstruktur in rekonfigurierbarer Hardware. Hierzu können analoge rekonfiguierbare Zellen in *Field Programmable Analog Arrays* (FPAA) oder die digitale Nachbildung in *Field Programmable Gate Arrays* (FPGA) verwendet werden. FPAAs sind eine vergleichsweise neue Entwicklung, die entweder als Transistorarray [5] oder mittels Analogzellen aufgebaut wird und vielversprechende Leistungsdaten aufweist [6]. Da noch keine Implementierungstools vorhanden sind, beschränkt sich deren Anwendung auf Experimentalaufbauten. Die Emulation des Verhaltens analoger Schaltungen in digitalen FPGAs ist gemäß des Nyquist-Kriteriums für ausreichend hohe Taktfrequenzen möglich [7]. Ansätze zur Abbildung der analogen Struktur sind als Übertragungsfunktion [8], Signalfluss-Blockbeschreibung [9] oder auf Komponentenebene durch den Wave-Digital-Filter-Ansatz [10] möglich.

In diesem Paper wird die Entwicklung der Hardware einer Emulator-in-the-Loop-Plattform (siehe Abbildung 1) zur digitalen Nachbildung von Mixed-Signal-Modulatoren mit digitalem Ausgang im Hinblick auf leistungselektronische Anwendungen vorgestellt. Die Emulation wird digital auf einem FPGA durchgeführt. Damit sollen die Nachteile der bisher etablierten Untersuchungsansätze mit Simulationen und physikalischer Implementierung überwunden werden.

II. Systemkonzept

Das System wird modular aufgebaut [11], um das Entwicklungsrisiko durch Teilprojekte zu reduzieren, Erweiterbarkeit sicherzustellen und die Flexibilität

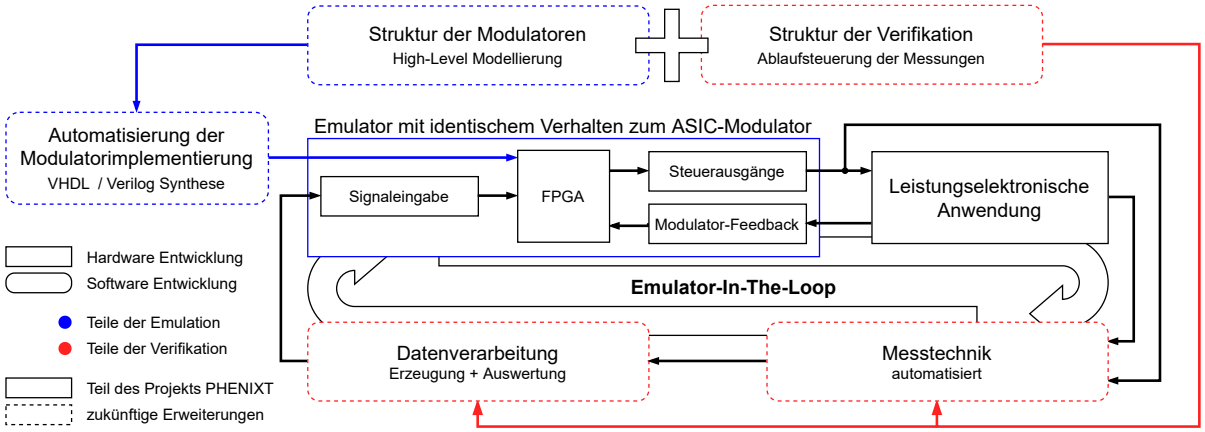


Abbildung 2. Gesamtsystemkonzept der Emulationsplattform zur automatisierten Erforschung von Modulatortopologien für die Leistungselektronik

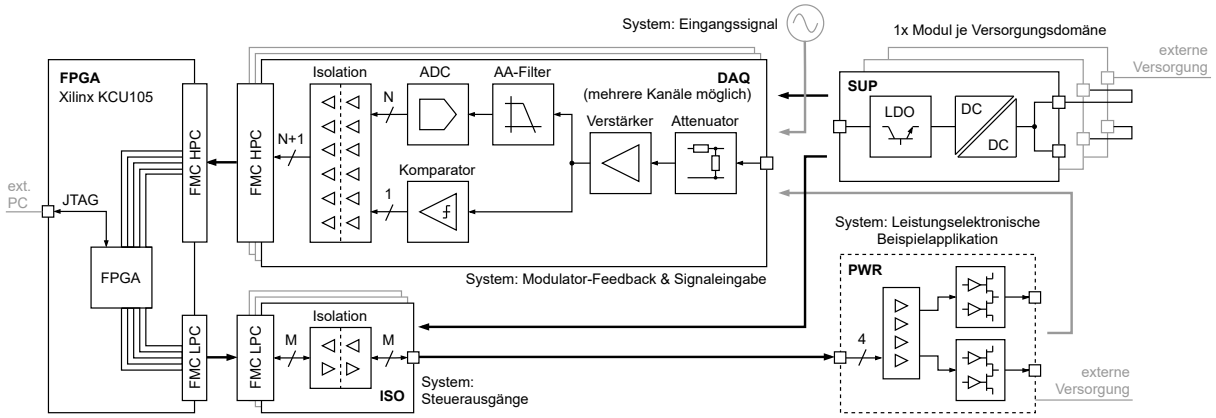


Abbildung 3. Partitionierungs- und Teilsystemkonzept für die zu implementierende Hardware der Emulationsplattform

zu erhöhen. Die geringere Integrationsdichte wird aufgrund der Anwendung als Forschungswerkzeug als nicht erheblich eingeschätzt. Als Vorlage dient das *"Hardware-in-the-Loop"*-Entwicklungsmodell für eingebettete Systeme, bei dem ein zu entwickelndes Hardwaresystem in einer Schleife mit einer von Laborgeräten simulierten Einsatzumgebung betrieben, vermessen und verifiziert wird [12]. Da im vorligenden Fall die Hardware durch eine Emulation ersetzt wird, wird der Begriff des *"Emulator-in-the-Loop"*-Systems verwendet (siehe Abbildung 2).

Die Emulationshardware (siehe Abbildung 3) besteht aus dem hochperformanten FPGA der Firma Xilinx (siehe [13]) mit Konfigurationsschnittstellen zur Implementierung der Modulatortopologie sowie digitalen Schnittstellen für die Ein- und Ausgabe von digitalen Signalen und analogen Schnittstellen zur Aufnahme analoger Eingangssignale. Alle Schnittstellen werden zum FPGA sowie zueinander galvanisch getrennt realisiert. Dies dient dem Schutz des FPGA und ermöglicht eine flexible Messung der Applikation. Die Spannungsversorgung wird aufgrund der Vielzahl an zur Verfügung zu stellenden galvanisch getrennten Spannungsdomänen und -ebenen in Modulbauweise von den Signalverarbeitungsplatinen getrennt realisiert.

Die analoge Schnittstelle entscheidet maßgeblich über die vom Emulationssystem maximal erreichbare Signalqualität. Diese wird den Designs von Oszilloskop-Frontends nachempfunden [14] [15] [16]. Da die Anforderungen vergleichbar sind. Anforderungen sind die zeitmäßige Auflösung als vom Modulator höchste fehlerfrei abgebildete Frequenz f_{emu} und die wertmäßige Auflösung, ausgedrückt durch die Kennzahl SINAD (engl. *"signal-to-interference-ratio including noise and distortion"* [17]), die das Verhältnis von allen erwünschten zu allen unerwünschten Signalanteilen angibt. Weitere Anforderungen sind in Tabelle I definiert.

Die "leistungselektronische Beispielapplikation" (siehe Abbildung 3) wird separat für die jeweils zu untersuchende Anwendung gestaltet und ist aufgrund des modularen Systemaufbaus einfach austauschbar. Mess-

Tabelle I
ANFORDERUNGEN DER SYSTEMEBENE AN DIE ZU IMPLEMENTIERENDE HARDWARE DER EMULATIONSPLATTFORM

Symbol	Beschreibung	min	typ	max	Einheit
SINAD	Signalqualität		80		dBFS
f_{Sig}	Signalfrequenz	0		2	MHz
Z_{in}	Eingangsimpedanz		1		MΩ
$t_{\text{pd,an}}$	Laufzeit Eingänge			100	ns
U_{in}	Eingangsspannung	±50			V

und Datenverarbeitungssysteme führen die Verifikation des emulierten Systems durch. Diese sollen von Softwarekomponenten zur automatischen Datenverarbeitung angesteuert werden. Teile dieser Aufgaben können vom FPGA selbst übernommen werden. Eine in der Hierarchie höher angesiedelte Modellierungsoberfläche soll mittels grafischer Methoden oder Hardwarebeschreibungssprachen die automatisierte Implementierung von Modulatortopologien über einen synthesefähigen Interpreter in einen implementierbaren Bitstream umsetzen. Diese Teilsysteme sind kein Bestandteil der vorliegenden Arbeit.

III. UMSETZUNG DER HARDWARE

Es werden die Definition der Frequenzverhältnisse an der analogen Schnittstellen sowie die Bereiche der analogen Signalvorverarbeitung und des analogen Attenuators betrachtet.

A. Frequenzverhältnisse

Um Aliasing zu vermeiden und dennoch hohe Signalfrequenzen abbilden zu können, wird für die Abtastung ein Anti-Aliasing-Filter definiert, da in typischen leistungselektronischen Signalen (Rechteck, Dreieck) Harmonische über $f_{\text{S}}/2$ auftreten. Die Abtastrate ist, aus Gründen der Realisierbarkeit, so gering wie möglich zu wählen. Dies hat eine geringe Filtergrenzfrequenz zur Folge. Je kleiner die Grenzfrequenz, desto mehr Oberwellen des Nutzsignals können nicht digital rekonstruiert werden. Die Grenzfrequenz und Ordnung des Filters werden so gewählt, dass eine minimale Durchlaufzeit und Signalverzerrung für ein vollständiges Verhindern von Alias-Fehlern erreicht wird.

Es wird folgendes Vorgehen gewählt (siehe dazu Abbildung 4): Es wird als Worst-Case-Eingangssignal von einer Trapezfunktion ausgegangen. Die Grenzfrequenz des Filters wird in den Alias-Bereich ausgedehnt, ohne das Aliasing auftritt. Die Hüllkurve des Spektrums einer Trapezfunktion ist monoton fallend und weist bei Erreichen der Nyquist-Frequenz $f_{\text{S}}/2$ eine gewisse Eigendämpfung auf. Der Antialiasing-Filter mit Grenzfrequenz $f_{\text{c}} < f_{\text{S}}/2$ bringt zusätzlich eine additive Dämpfung ein. Die am ADC gemessene Amplitude der Harmonischen ist kleiner als im Spektrum des Trapezsignals oder im Bode-Diagramm des Filters. Mit

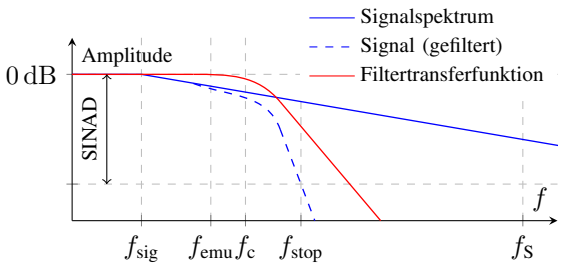


Abbildung 4. Verhältnisse der Arbeits- und Grenzfrequenzen in der analogen Vorverarbeitung gemäß des beschriebenen Auslegungsverfahrens

der Definition einer Passband-Abweichung von ΔG wird die Grenzfrequenz so festgelegt, dass bei f_{emu} die maximal spezifizierte Abweichung gegenüber $f = 0\,\text{Hz}$ auftritt. Das Stoppband beginnt bei der Frequenz f_{stop}, bei der das Signalspektrum des Worst Case Eingangssignals multipliziert mit der Filtertransferfunktion das SINAD erreicht. Die Abtastfrequenz ist $f_{\text{S}} = 2f_{\text{stop}}$.

B. Signalvorverarbeitung

Da die Wahl der Abtastrate und der Filterparameter wie oben beschrieben in Verbindung stehen, wird eine zweistufige Auslegung durchgeführt, um die Designreserven möglichst vollständig auszuschöpfen. Im ersten Schritt wird gemäß des vorgestellten Auslegungsverfahrens eine minimale Abtastfrequenz mit den Anforderungen der Anwendung gefunden. Im zweiten Schritt wird die Grenzfrequenz bis zum Maximum des gewählten ADC erhöht.

Ein Bessel-Verhalten mit konstanter Gruppenlaufzeit ist aufgrund des notwendigen idealen Rechteckübertragungsverhaltens alternativlos. Die Ordnung 7 wurde als Kompromiss zwischen Flankensteilheit und Gruppen-laufzeit sowie Umsetzungsaufwand definiert. Der Filter wird differenziell aufgebaut, da die erhöhe Störfestigkeit bei geringen Spannungen vorteilhaft wirkt. Ein passiver Aufbau als LC-Filter in Leitertopologie wurde gegenüber einer aktiven Kaskade mit Operationsverstärkern aufgrund deren hoher Kosten für große Ordnungen bevorzugt. In Simulationen konnte kein entscheidender Vorteil eines solchen aktiven Filters nachgewiesen werden.

Mit einer Passband-Abweichung von $\Delta G = \pm 1\,\%$ und der Worst-Case-Annahme $f_{\text{sig}} = f_{\text{emu}}$ ergibt sich eine minimale Abtastfrequenz von $f_{\text{S}} \geq 98{,}34\,\text{MHz}$ (bei $f_{\text{c}} = 11{,}55\,\text{MHz}$). Zum Erreichen des System-SINAD nach $\text{SQNR} = 6{,}02 \cdot B + 1{,}76\,\text{dB}$ sind $\geq 13\,\text{Bit}$ erforderlich. Zuzüglich Auslegungstoleranzen wird der 16 Bit ADC LTC2209 [18] mit 160 MSPS ausgewählt. Die Grenzfrequenz kann auf $f_{\text{c}} = 17{,}7\,\text{MHz}$ erhöht werden.

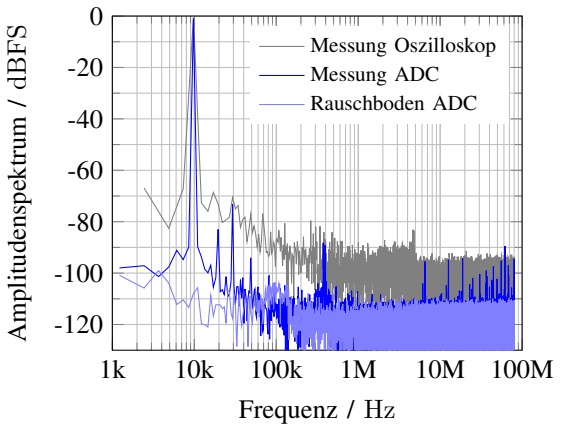


Abbildung 5. Vergleich der Amplitudenspektren des Oszilloskops Teledyne LeCroy HDO6104 und des implementierten Datenakquisesystems, $f_{\text{in}} = 9{,}7656\,\text{kHz}$, $A_{\text{in}} \approx -1\,\text{dBFS}$ differenziell

C. Attenuator

Zur Aufnahme verschiedener Eingangsspannungen wurde eine Anpassung des Spannungsbereichs des ADC an die Applikation vorgenommen. Ähnlich eines Oszilloskop-Frontends [19] wird eine Kaskade frequenzkompensierter Spannungsteiler mit unterschiedlichen Teilerverhätnissen und jeweils gleichen Eingangsimpedanzen eingesetzt. Durch den Einsatz von Tastköpfen als verlustbehaftete Leitungen weden Leitungsreflexionen trotz der Entfernung der Messung von der Messstelle unterdrückt. Die Entkopplung des Attenuators erfolgt durch einen Transimpendanzverstärker mit Feldeffekttransistor-Eingang. Es wurden Teilungsfaktoren von $1, 2, 5, 10$ umgesetzt, die mit einem $1:10$ Tastkopf einen Eingangsspannungsbereich bis $225\,\text{Vpp}$ ermöglichen. Pro differenziellem Kanal werden zwei Attenuatoren einsetzt, um differenzielle Messungen zu ermöglichen.

IV. Messergebnisse

Die Signalqualität der analogen Schnittstelle wurde mit einem kommerziellen Oszilloskop verglichen (siehe Abbildung 5). Die von der umgesetzten Schaltung erreichte Signalqualität liegt bei $65{,}48\,\text{dB}$ für den vollständigen Frequenzbereich von $0\,\text{Hz}$ bis $80\,\text{MHz}$, der störungsfreie Dynamikbereich bei $-73{,}17\,\text{dB}$, der Rauschboden liegt bei $-99{,}03\,\text{dB}$. Das zum Vergleich herangezogene Digitaloszilloskop LeCroy HDO6104 zeigt eine um $10{,}78\,\text{dB}$ schlechtere Signalqualität. Das Übertreffen eines kommerziellen Oszilloskops zeigt die Relevanz der umgesetzten Hardware. Der Frequenzgang des Kanals zeigt eine maximale Abweichung von $0{,}376\,\text{dB}$ (bis $11{,}37\,\text{MHz}$), die Grenzfrequenz liegt bei $24{,}15\,\text{MHz}$ (siehe Abbildung 6).

Die Verzögerungszeit von der Tastkopfspitze bis zum Vorliegen des Registerwerts im FPGA beträgt $90{,}06\,\text{ns}$ (siehe Abbildung 7). Eine Modulatorimplementation kann zuzüglich Verarbeitungszeit erst danach eine weitere Entscheidung über den Ausgang treffen.

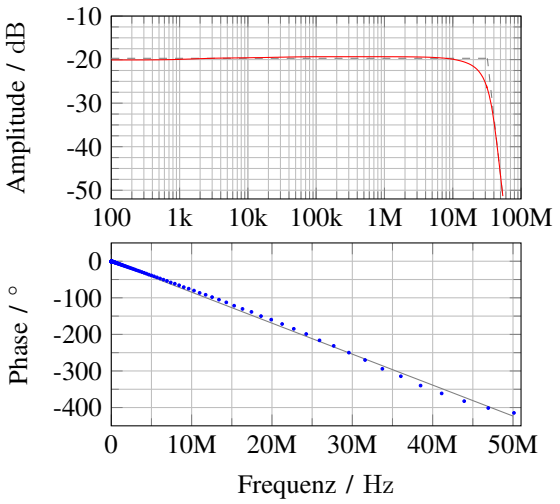


Abbildung 6. Amplituden- und Phasengang der vollständigen analogen Signalverarbeitung einschließlich Tastköpfe, $A_{\text{in}} \approx -1\,\text{dBFS}$ differenziell

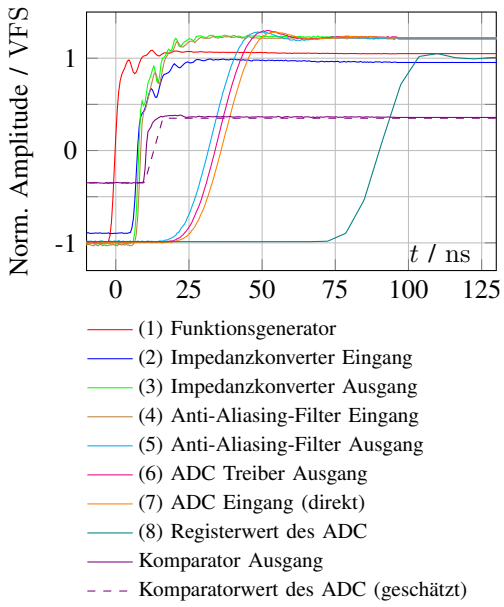


Abbildung 7. Durchlaufzeiten für verschiedene Punkte entlang der analogen Signalverarbeitungskette, normiert auf die jeweilige Full-Scale-Range, $U_{\text{in}} = 10\,\text{Vpp}$ differenziell

V. Fazit

Es wurden das Konzept einer Plattform für die Emulation von Mixed-Signal-Schaltungen für Anwendungen in der Leistungselektronik, sowie der Entwurf und die Verifikation einer dafür benötigten Hardware vorgestellt. Die erzielten Spezifikationen ermöglichen eine Emulation bis zu einer Frequenz von $11{,}37\,\text{MHz}$ mit einer um $10{,}78\,\text{dB}$ besseren Signalqualität gegenüber einem kommerziell erhältlichen Oszilloskop. Die Implementierung eines beispielhaften closed-loop-Modulators wurde nachgewiesen. Damit kann die Charakterisierung von Modulatoren in Zukunft automatisiert erfolgen, wodurch neue Untersuchungsansätze eröffnet werden.

In einem weiteren Schritt ist geplant, das System um Softwarekomponenten zur automatisierten Synthese der Modulatoren und zur Messgerätesteuerung zu ergänzen.

LITERATURVERZEICHNIS

[1] WERNER, MARTIN: *Nachrichtentechnik. Eine Einführung für alle Studiengänge*. 7. Auflage, Wiesbaden, 2010.

[2] KUO, CHIEN-HUNG, LIN, SHENG-CHI: *A delta-sigma modulator-based class-D amplifier*. 2016 IEEE 5th Global Conference on Consumer Electronics, Seiten 1-2, Kyoto, 2016.

[3] DE LA ROSA, JOSE M.: *Sigma-Delta Converters. Practical Design Guide*. 2. Auflage, Hoboken, 2018.

[4] WOLFER TOBIAS, HENNIG, ECKARD: *An Active Feedback Coefficient Tuning Technique for Compensating Time-Constant Variations in Continuous-Time Delta-Sigma Modulators*. 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), Seiten 17-20, Penang, 2021.

[5] SUDA, NAVEEN ET. AL.: *A 65 nm Programmable ANalog Device Array (PANDA) for Analog Circuit Emulation* In: *IEEE Transactions on Circuits and Systems I: Regular Papers*. Vol. 63, Nr. 2, Seiten 181-190, Tempe, Santa Clara, 2016.

[6] ANSARI, TARAB,YASIN, MOHD. YUSUF: *High frequency FPAA Design using OTA in 45nm CMOS Technology*. 2018 4th International Conference on Computing Communication and Automation (ICCCA), Seiten 1-5, Greater Noida, 2018.

[7] MEYER, MARTIN: *Signalverarbeitung. Analoge und digitale Signale, Systeme und Filter*. 9. Auflage, Wiesbaden, 2021.

[8] HERBST, STEVEN ET. AL.: *An Open-Source Framework for FPGA Emulation of Analog/Mixed-Signal Integrated Circuit Designs*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, o.O., 2021.

[9] TERTEL, PHILIPP , HEDRICH, LARS: *Real-time emulation of block-based analog circuits on an FPGA*. In: *14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. Seiten 1-4, Frankfurt am Main, 2017.

[10] WU, WEI ET. AL.: *Wave digital filter based analog circuit emulation on FPGA*. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. Seiten 1286-1289, Los Angeles, 2016.

[11] DOBOLI, ALEX, CURRIE, EDWARD H.: *Introduction to Mixed-Signal, Embedded Design*. Verlag, New York, 2011.

[12] GESSLER, RALF: *Entwicklung Eingebetteter Systeme. Vergleich von Entwicklungsprozessen für FPGA- und Mikroprozessor-Systeme Entwurf auf Systemebene*. Springer Vieweg Verlag, Wiesbaden, 2014.

[13] XILINX INC. (HRSG.): *KCU105 Board. User Guide*. UG917, o.O., 2019.

[14] BHAT, ROHIT, TEXAS INSTRUMENTS INC.: *50-Ohm 2-GHz Oscilloscope Front-end Reference Design*. Design Guide TIDA-00826, Dallas, 2015.

[15] TEKTRONIX INC.: *Tektronix 465 Oscilloscope Service Instruction Manual*. Beaverton, o.J.

[16] TEKTRONIX INC.: *2205 Oscilloscope Service*. Service Manual, Beaverton, 1988.

[17] KESTER, WALT, ANALOG DEVICES INC.: *Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor*. Mini Tutorial MT-003, Rev. A., o.O., 2009.

[18] LINEAR TECHNOLOGY CORPORATION (HRSG.): *LTC2209. 16-Bit, 160Msps ADC*. Data Sheet, Milpitas, 2007.

[19] ROACH, STEVE: *Signal Conditioning in Oscilloscopes and the Spirit of Invention*. In WILLIAMNS, JIM (HRSG.): *The Art and Science of Analog Circuit Design*. Seiten 65-84, The EDN Series for Design Engineers, Butterworth-Heinemann Verlag, Boston, 1998.

Philipp Czerwenka erhielt den akademischen Grad B.Eng. im Studiengang Elektrotechnik von der Dualen Hochschule Baden-Württemberg Mannheim im Jahr 2020 und den M.Sc. im Studiengang Leistungs- und Mikroelektronik von der Hochschule Reutlingen im Jahr 2022. Aktuell arbeitet er dort als wissenschaftlicher Mitarbeiter am Lehr- und Forschungszentrum Electronics & Drives im Bereich Leistungsektronik und Antriebstechnik.

Tobias Wolfer erhielt den akademischen Grad B.Eng. im Studiengang Elektrotechnik von der Dualen Hochschule Baden-Württemberg Stuttgart im Jahr 2016 und den M.Sc. im Studiengang Leistungs- und Mikroelektronik an der Hochschule Reutlingen im Jahr 2019. Aktuell arbeitet er dort als wissenschaftlicher Mitarbeiter und Doktorand am Lehr- und Forschungszentrum Electronics & Drives im Bereich Modulatoren für schaltende Leistungsverstärker.

Eckhard Hennig erhielt den akademischen Grad Dipl.-Ing. von der Technischen Universität Braunschweig im Jahr 1994 und den Dr.-Ing. an der Universität Kaiserslautern im Jahr 2000. Er ist Professor für Integrierte Schaltungstechnik an der Hochschule Reutlingen. Seine Forschungsbereiche umfassen Low-Power-CMOS-Schatungstechnik für Smart-Sensor-Anwendungen und Electronic Design Automation für Analog/Mixed-Signal-Schaltungen.

Hochschule Reutlingen
Reutlingen University

Asynchroner zeitkontinuierlicher Delta-Sigma-Modulator-ASIC für GaN-basierte HiFi-Klasse-D-Audio-Verstärker

Andreas Brunner, Philipp Czerwenka, Tobias Wolfer, Eckhard Hennig

***Zusammenfassung*—In diesem Paper wird ein HiFi-Klasse-D-Verstärker für Audiosignale im Frequenzbereich zwischen 20 Hz und 20 kHz präsentiert. Das System verwendet pro Stereo-Kanal zwei unabhängig betriebene Modulatoren zur Ansteuerung zweier Halbbrücken, die einen differenziellen Ausgangsfilter treiben. Die asynchronen, zeitkontinuierlichen Delta-Sigma-Modulatoren werden in einem 0,35 µm-Analog/Mixed-Signal-CMOS-ASIC umgesetzt. Die maximale Oszillationsfrequenz ist mit 4 Bit zwischen 500 kHz und 2 MHz und die interne Totzeit-Generierung mit 2 Bit zwischen 15 ns und 32 ns einstellbar. Für die 48 V-Leistungselektronik (LE) kommen Gallium-Nitrid (GaN)-Halbbrücken mit monolithisch integrierten Gatetreibern zum Einsatz. Im Vergleich zu Silizium-Technologien können die Kosten und das Bauvolumen des passiven Leistungsfilters reduziert werden. Es wird eine Ausgangsleistung von 100 W pro Kanal an einer Last von 4 Ω bis 8 Ω bei einer Gesamtsignalqualität (THD+N) von bis zu 56 dB erreicht.**

***Schlüsselwörter*—ASIC, Delta-Sigma, Modulator, Gallium-Nitrid (GaN), Leistungselektronik**

I. Einleitung

Audioverstärker werden zur Verstärkung von Audiosignalen und zum Treiben der niederimpedanten Lautsprecher verwendet (4 Ω bis 8 Ω). Dabei ist eine möglichst verzerrungsarme Verstärkung im hörbaren Frequenzbereich von 20 Hz bis 20 kHz notwendig. Im Vergleich zu traditionellen Klasse-A- oder Klasse-B-Verstärkern, weist der Klasse-D-Verstärker eine geringere Baugröße und eine höhere Energieeffizienz auf [1], [2], [3], [4]. In Abbildung 2 ist das Blockschaltbild eines Klasse-D-Verstärkers dargestellt. Das niederfrequente Nutzsignal (Audiosignal) wird durch einen Eingangsfilter (Anti-Aliasing-Filter) bandbegrenzt und von einem Modulator in ein Digitalsignal konvertiert. Die Information des Nutzsignals ist hier in Frequenz und Tastgrad kodiert. Die Verstärkung des Digitalsignals erfolgt durch die Endstufen (Halb- oder Vollbrücke). Im Vergleich zur Halbbrücke verdoppelt sich bei der Verwendung einer Vollbrücke der Spannungshub über der Last, was zu einer viermal höheren Ausgangsleistung führt [5]. Klasse-D-Verstärker verwenden die Leistungstransistoren der Endstufen als Schalter und verursachen somit geringe Verluste. Leistungstransistoren aus Wide-Bandgap-Halbleitermaterialien werden wegen den kurzen Schaltzeiten und geringen Schaltverlusten in der Leistungselektronik immer beliebter [6]. Zur Demodulation des verstärkten Nutzsignals und zur Unterdrückung der Trägerfrequenz wird ein Tiefpassfilter verwendet. Durch die höheren Schaltfrequenzen können die Leistungsfilter kleiner dimensioniert werden [7]. Dies steigert die Kosteneffizienz und die volumetrische sowie gravimetrische Leistungsdichte des Gesamtsystems.

Andreas Brunner, andreas.brunner@reutlingen-university.de, Philipp Czerwenka, philipp.czerwenka@reutlingen-university.de, Tobias Wolfer, tobias.wolfer@reutlingen-university.de, Eckhard Hennig, eckard.hennig@reutlingen-university.de. Hochschule Reutlingen, Lehr- und Forschungszentrum Electronics & Drives, Oferdinger Straße 50, 72768 Rommelsbach.

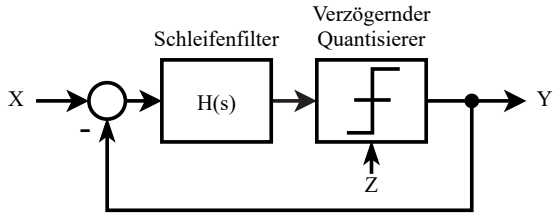


Abbildung 1. Asynchroner, zeitkontinuierlicher Delta-Sigma-Modulator.

Neben Pulsweiten-Modulatoren (PWM) werden im Audiobereich asynchrone Delta-Sigma-Modulatoren für die Analog-Digital-Wandlung genutzt [8]. Das Blockschaltbild eines asynchronen, zeitkontinuierlichen Delta-Sigma-Modulators (Abbildung 1) besteht aus einem Schleifenfilter, Quantisierer sowie einer Rückkopplungsschleife. Die hohe Signalauflösung wird durch Überabtastung des analogen Eingangssignals erreicht. Zur Reduktion der analogen Komplexität des Modulators wird damit die Auflösung der Amplitudenquantisierung gegen eine höhere Zeitauflösung eingetauscht [9], [10]. Die Störübertragungsfunktion verschiebt den Großteil der Leistung des Quantisierungsrauschens aus dem Nutzband heraus (noise shaping) [8], [11] und erhöht die Signalqualität im Audioband.

Asynchrone Modulatoren benötigen keinen externen Takt und zeichnen sich durch einen geringen Energiebedarf aus [12], [13]. Sie besitzen zudem eine inhärente Stabilität [6] und Anti-Alias-Filterung [14].

Folgend werden die Anforderungen, das System und Simulationsergebnisse (Abschnitt II) des asynchronen zeitkontinuierlichen Delta-Sigma-Modulator gezeigt. Es

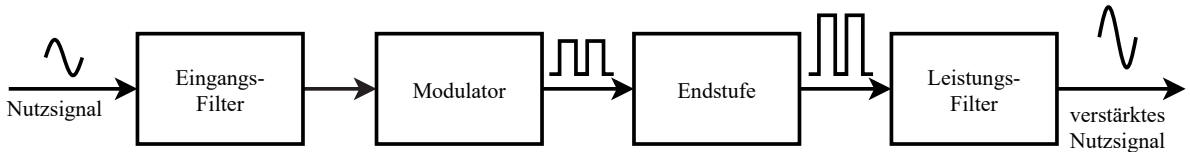


Abbildung 2. Blockschaltbild eines Klasse-D-Verstärkers.

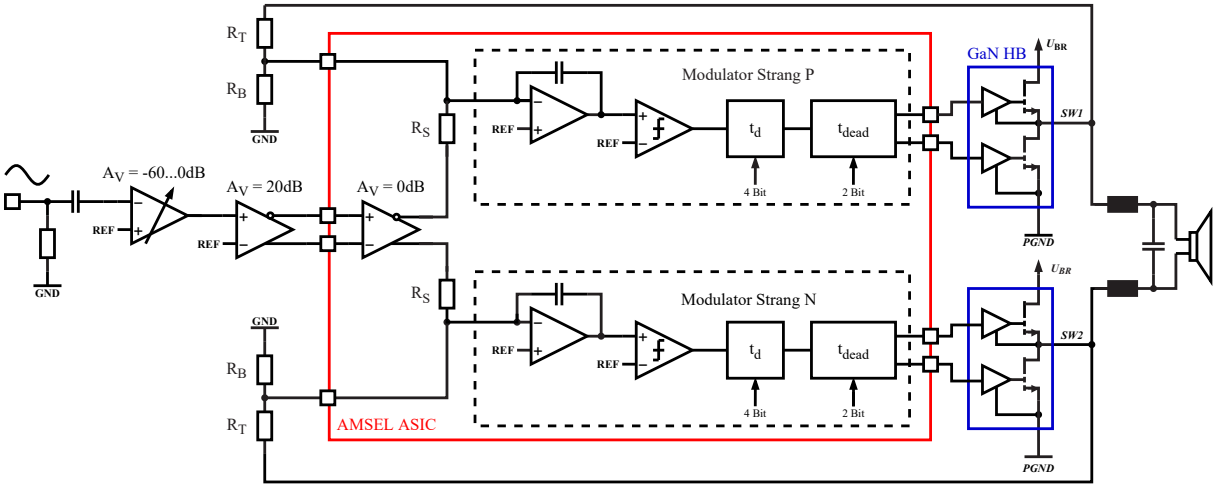


Abbildung 3. Blockschaltbild für einen Kanal des HiFi-Klasse-D-Audioverstärkers.

folgen die Beschreibungen der Implementierung sowie der Messergebnisse des Verstärkers (Abschnitt III). Das Paper schließt mit einem Fazit sowie Ausblick ab (Abschnitt IV).

II. Systemkonzept

A. Anforderungen

In Tabelle I sind die Anforderungen des Audioverstärkers aufgelistet.

Tabelle I
Anforderungen des Audioverstärkers.

Parameter	Wert	Einheit
Versorgungsspannung IC	3,3	V
Oszillationsfrequenz IC	0,5 - 2	MHz
Pins IC-Package	max. 44	-
Versorgungsspannung LE	48	V
Audio-Ausgangsleistung	min. 100	W
THD+N	max. 0,1	%

B. Modellierung

Abbildung 3 zeigt das Blockschaltbild für einen Kanal des Audioverstärkers. Das Audiosignal wird zuerst durch einen diskreten Bandpassfilter mit einstellbarer Verstärkung bandbegrenzt, vorverstärkt und von single-ended auf ein differenzielles Signal gewandelt. Diese Signale werden auf die beiden Stränge des Kanals im ASIC aufgeteilt. Der ASIC enthält zwei Kanäle für den Stereo-Betrieb mit jeweils zwei unabhängigen asynchronen, zeitkontinuierlichen Delta-Sigma-Modulatoren. Ein Modulator-Strang besteht aus der Kette von Integrator, einem Komparator, einer Verzögerungskette und einer Totzeitlogik. Die Regelschleife wird über ein diskretes Widerstandsnetzwerk geschlossen.

Die Verzögerungskette besteht aus einer geraden Anzahl von hintereinandergeschalteten Invertern. Über 4 Bit wird extern die Anzahl der Inverterketten eingestellt und variieren somit die maximale Oszillationsfrequenz des Modulators zwischen $500\,\mathrm{kHz}$ und $2\,\mathrm{MHz}$. Die Totzeitlogik erzeugt aus dem verzögerten Signal des Komparator-Ausgangs die Ansteuersignale für den High- und Low-Side-Schalter der Endstufe. Die Totzeit wird extern mit 2 Bit zwischen $15\,\mathrm{ns}$ und $32\,\mathrm{ns}$ eingestellt werden.

Für die Endstufen werden die Halbbrücken-Bausteine vom Typ EPC2152 mit GaN-HEMTs und monolithisch integrierten Gatetreibern verwendet [15]. Diese werden mit einer Brückenspannung von $U_{\mathrm{BR}} = 48\,\mathrm{V}$ versorgt. Die zwei Schaltknoten eines Kanals sind über ein differenzielles LC-Tiefpassfilter mit der Last (Lautsprecher) verbunden.

Für die Systembetrachtung werden zuerst die Ströme am Integratorknoten analysiert. Abbildung 5 zeigt die detaillierte Beschaltung des Integrators mit der Spannung am Schaltknoten U_{SW}, der Ausgangsspannung des voll-differenziellen Verstärkers U_{FDA} und der Referenzspannung $U_{\mathrm{REF}} = 1{,}65\,\mathrm{V}$. Folgend werden zwei Zustände betrachtet. Im Zustand 1 liegt der Schaltknoten SW auf hohem Potential, d.h. $U_{\mathrm{SW}} = U_{\mathrm{BR}}$. Im

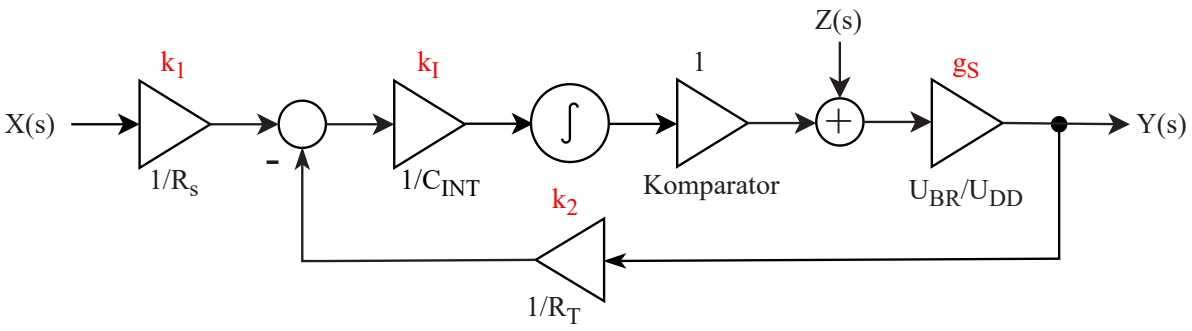


Abbildung 4. Kleinsignalmodell des asynchronen zeitkontinuierlichen Delta-Sigma-Modulators.

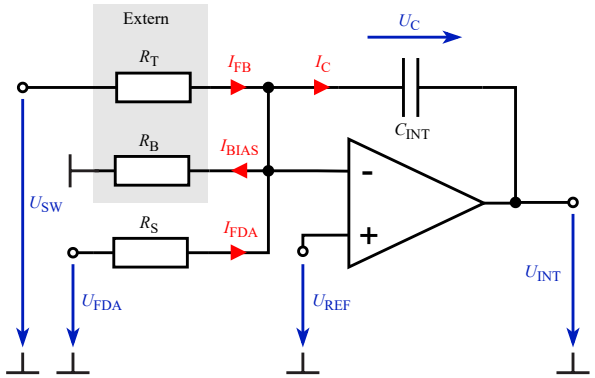


Abbildung 5. Detaillierte Beschaltung des Integrators.

Zustand 2 liegt der Schaltknoten SW auf niedrigem Potential, d.h. $U_{\mathrm{SW}} = 0\,\mathrm{V}$. Im Zustand 1 steigt die Ausgangsspannung des Komparators U_{INT} an und der Strom I_{C} in die Integratorkapazität C_{INT} ist

$$I_{\mathrm{C},1} = \frac{U_{\mathrm{BR}} - U_{\mathrm{REF}}}{R_{\mathrm{T}}} - \frac{U_{\mathrm{REF}}}{R_{\mathrm{B}}} + \frac{\Delta U_{\mathrm{FDA}}}{R_{\mathrm{S}}}. \quad (1)$$

Im Zustand 2 sinkt U_{INT} und der Strom in die Integratorkapazität ist

$$I_{\mathrm{C},2} = -U_{\mathrm{REF}} \left(\frac{1}{R_{\mathrm{T}}} + \frac{1}{R_{\mathrm{B}}} \right) + \frac{\Delta U_{\mathrm{FDA}}}{R_{\mathrm{S}}}. \quad (2)$$

Aus dem Blockschaltbild des Verstärkers (Abbildung 3) kann das Kleinsignalmodell von Abbildung 4 abgeleitet werden. Daraus ergibt sich ein Tiefpass-Verhalten für die Signalübertragungsfunktion (STF, Signal Transfer Function)

$$STF(s) = \frac{Y(s)}{X(s)} = \frac{k_1}{k_2} \frac{1}{1 + s\,(g_{\mathrm{s}}\,k_{\mathrm{I}}\,k_2)^{-1}} \quad (3)$$

sowie ein Hochpass-Verhalten für die Störübertragungsfunktion (NTF, Noise Transfer Function)

$$NTF(s) = \frac{Y(s)}{Z(s)} = \frac{1}{k_{\mathrm{I}}\,k_2} \frac{s}{1 + s\,(g_{\mathrm{s}}\,k_{\mathrm{I}}\,k_2)^{-1}}. \quad (4)$$

Tabelle II
VERWENDETE BAUTEILPARAMETER DES MODULATORS.

Parameter	Wert	Einheit
R_{T}	300	kΩ
R_{B}	22	kΩ
R_{S}	15	kΩ
C_{INT}	124	pF

Somit kann das folgende Gleichungssystem zur Dimensionierung der Freiheitsgrade R_{T}, R_{B}, R_{S} und C_{INT} des Modulators aufgestellt werden:

$$A_{\mathrm{v}} = \frac{R_{\mathrm{T}}}{R_{\mathrm{S}}} \quad (5)$$

$$I_{\mathrm{C,max}} = C_{\mathrm{INT}} \frac{|\Delta\,U_{\mathrm{INT}}|}{t_{\mathrm{d,max}}} \quad (6)$$

$$|I_{\mathrm{C},1}| = |I_{\mathrm{C},2}|_{\Delta U_{\mathrm{FDA}}=0\,\mathrm{V}} =: I_{\mathrm{C},0} \quad (7)$$

$$|I_{\mathrm{C},1}|_{\Delta U_{\mathrm{FDA}}=max} = I_{\mathrm{C,max}} \quad (8)$$

Nach dem Festlegen der Systemverstärkung $A_{\mathrm{V}} = 20$, des maximalen Ausgangsstroms des Operationsverstärkers $|\Delta I_{\mathrm{INT}}| < 500\,\mu\mathrm{A}$, des maximalen Integratorhubs $|\Delta U_{\mathrm{INT}}| < 600\,\mathrm{mV}$ und für den ungünstigsten Fall von $t_{\mathrm{d,max}} = 500\,\mathrm{ns}$ kann das Gleichungssystem eindeutig gelöst werden. In der Tabelle II sind die resultierenden Bauteilparameter aufgelistet.

C. Simulation

In Abbildung 6 ist das ideale Simulink-Simulationsmodell des Audioverstärkers mit den Parametern aus Tabelle II dargestellt. Als Eingangssignal wird ein Sinus mit einer Frequenz von $f = 1\,\mathrm{kHz}$ und einer Amplitude von $1\,\mathrm{V}$ (Full Scale) verwendet. Um eine maximale Oszillationsfrequenz des Modulators von $1\,\mathrm{MHz}$ zu erhalten, wird das Verzögerungsglied auf $t_{\mathrm{d}} = 250\,\mathrm{ns}$ eingestellt. In Abbildung 7 sind die Ergebnisse der Systemsimulation ersichtlich. Bei einer Eingangsspannung $V_{\mathrm{in}} = 0\,\mathrm{V}$ ist der Tastgrad der beiden Schaltknoten $50\,\%$. Dies

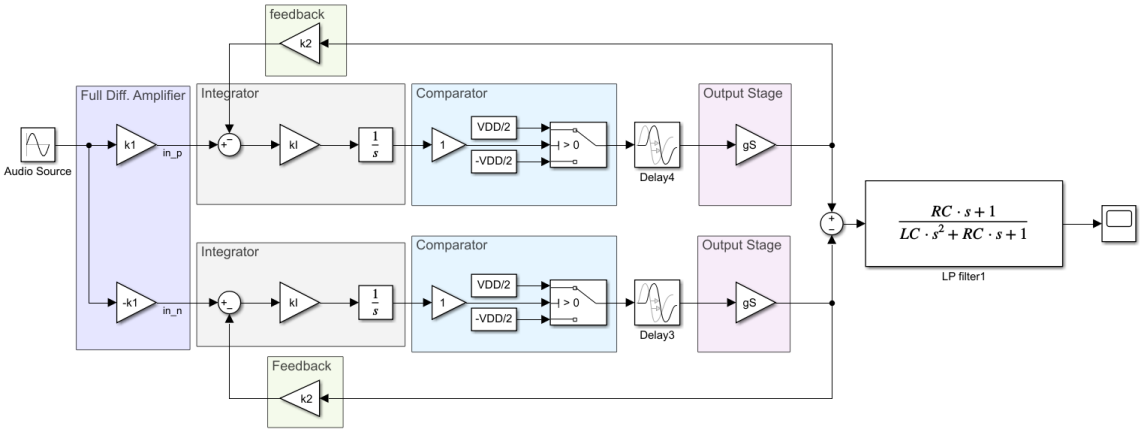


Abbildung 6. Simulink-Simulationsmodell für einen Kanal des HiFi-Klasse-D-Audioverstärkers.

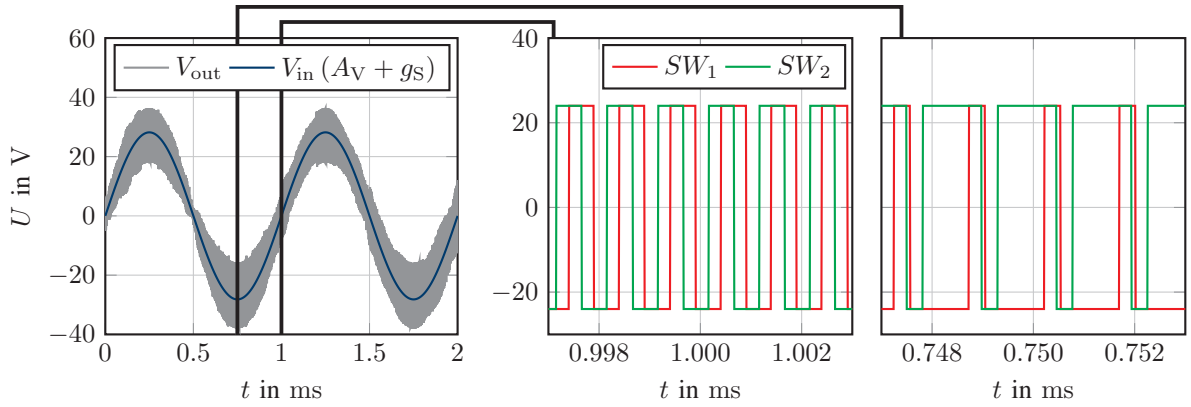


Abbildung 7. Simulationsergebnisse der Ein- und Ausgangsspannung des Verstärkers sowie die Spannungsverläufe der Schaltknoten bei $-1\,\mathrm{V}$ und $0\,\mathrm{V}$ Eingangsspannung im Verlauf eines Sinus-Eingangssignals.

führt differenziell zu einer Ausgangsspannung von $0\,\mathrm{V}$. Zudem ist dort die Oszillationsfrequenz maximal. Bei der geringsten Eingangsspannung $V_{in} = -1\,\mathrm{V}$ ist der Tastgrad des unteren Schaltknotens (SW_2) maximal und der Tastgrad des oberen Schaltknotens (SW_1) minimal, was zur minimalen Ausgangsspannung führt. Zudem ist in diesem Arbeitspunkt die Oszillationsfrequenz am geringsten. In Abbildung 8 ist die spektrale Leistungsdichte der Ausgangsspannung ersichtlich. Für das Signal-zu-Rausch-Verhältnis (THD+N, Total Harmonic Distortion + Noise) ergibt sich daraus ein Wert von $58{,}3\,\mathrm{dB}$.

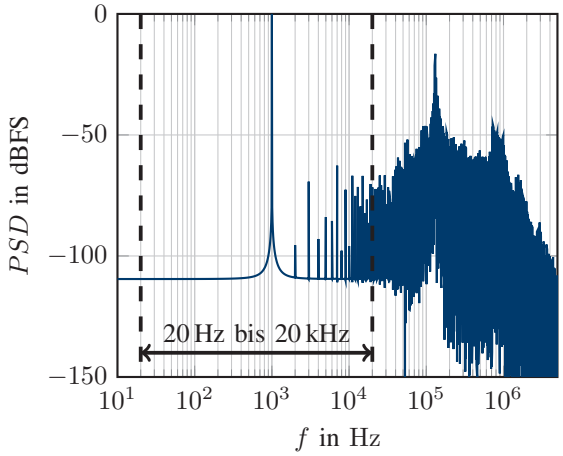


Abbildung 8. Simulierte spektrale Leistungsdichte des Audio-Ausgangssignals.

III. Implementierung und Messergebnisse

Für die Implementierung des ASICs wurde der $3{,}3\,\mathrm{V}$ - $0{,}35\,\mu\mathrm{m}$ Analog/Mixed-Signal-CMOS-Prozess XH035 von X-FAB Semiconductor Foundries GmbH verwendet. Die Schaltung wurde sowohl auf der Basis von Full Custom Design als auch mit Hilfe von X-FAB-Analog-IP-Blöcken erstellt. Der ASIC beinhaltet ca. 11.000 Transistoren, umfasst eine Fläche von $5{,}78\,\mathrm{mm}^2$ (Abbildung 10) und wurde in einem JLCC-Gehäuse mit 44 Pins gebondet.

Zur Vermessung des Audioverstärkers wurde ein vierlagiges Testboard ($140\,\mathrm{mm} \times 185\,\mathrm{mm}$) mit aufsteckbaren GaN-FET-Endstufen realisiert (Abbildung 11).

In Abbildung 9 ist die Gesamtsignalqualität des Audioverstärkers für eine $4\,\Omega$-Last über der Aus-

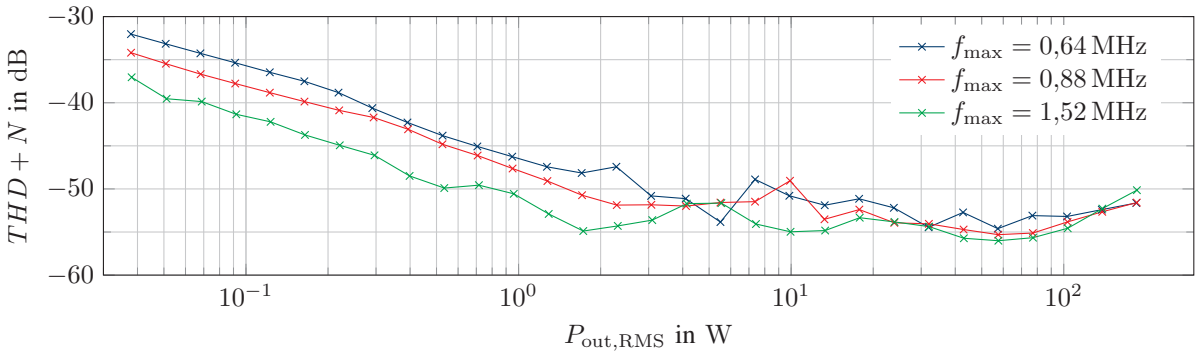


Abbildung 9. Gesamtsignalqualität des Audioverstärkers für eine 4 Ω-Last und verschiedene Schaltfrequenzen über der Ausgangsleistung.

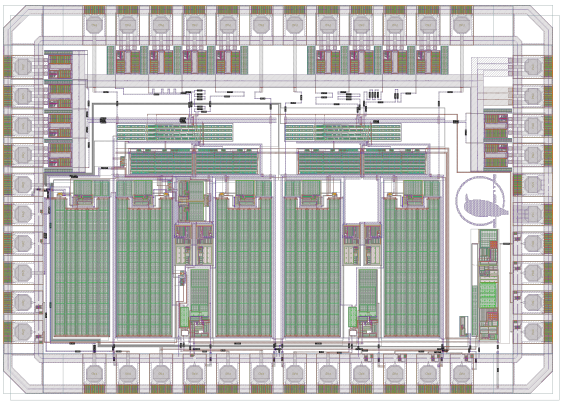


Abbildung 10. ASIC-Layout, 0,35 µm-CMOS-Prozess, 2,83 mm × 2,04 mm.

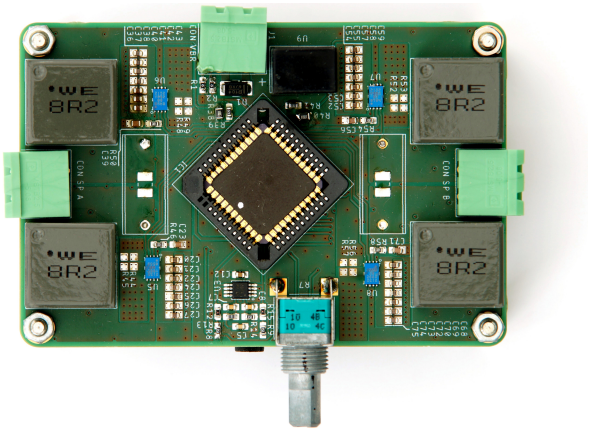


Abbildung 12. Kompaktes Verstärkerboard HiFi-Klasse-D-Verstärker, 60 mm × 90 mm.

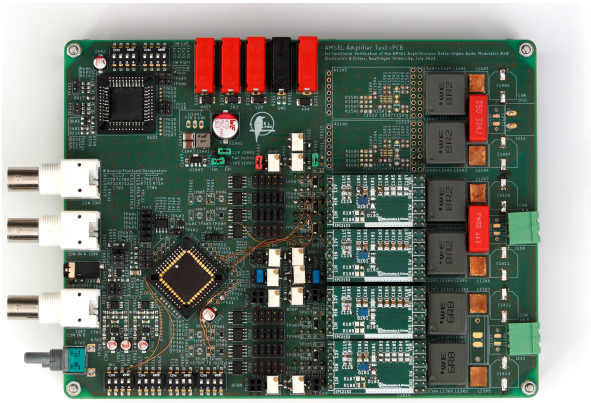


Abbildung 11. Testboard HiFi-Klasse-D-Verstärker, 140 mm × 185 mm.

gangsleistung P_{out} und verschiedenen Schaltfrequenzen ersichtlich. Als Eingangssignal wird ein Sinus mit $f = 1\,\text{kHz}$ verwendet. Mit steigender Ausgangsleistung wird das Rauschen des Modulators im Verhältnis zum Signalpegel geringer und die Signalqualität steigt. Bei Ausgangsleistungen über 100 W wird der zulässige Aussteuerbereich des Modulators verlassen und die Signalqualität sinkt. Die maximale Signalqualität (THD+N) wird bei ca. 57 W Ausgangsleistung und einer Schaltfrequenz von 1,52 MHz mit 56 dB erreicht. Für Ausgangsleistungen zwischen 0,8 W und 100 W ist THD+N größer als 50 dB. Für kleine Leistungen ($P_{\text{out}} < 10\,\text{W}$) steigt die Signalqualität mit steigender Oszillationsfrequenz des Modulators und ist bei 1,52 MHz um bis zu 7 dB besser als bei 0,64 MHz. Für höhere Leistungen beträgt die Signalverbesserung durchschnittlich 1,6 dB.

IV. Fazit und Ausblick

Es wurden das Systemkonzept, das ASIC-Design und die GaN-basierte Leistungselektronik eines HiFi-Klasse-D-Audio-Verstärkers mit asynchronem zeitkontinuierlichen Delta-Sigma-Modulator vorgestellt. Die erzielten Spezifikationen ermöglichen eine Musikwiedergabe auf HiFi-Qualität mit THD+N $> 50\,\text{dB}$ bei $2 \cdot 100\,\text{W}$ Ausgangsleistung. Durch die hohe Schaltfrequenz von bis zu 2 MHz ist es möglich, den Bauraum für Filterkomponenten zu reduzieren und die Qualität eines Klasse-AB-Verstärkers mit der Größe und der Effizienz eines konventionellen Klasse-D-Verstärkers zu vereinen.

Zurzeit wird an einem kompakten Design des Audioverstärkers für dieselben Spezifikationen gearbeitet. Diese Leiterplatte besitzt eine Größe von 60 mm × 90 mm und ist in Abbildung 12 dargestellt.

LITERATURVERZEICHNIS

[1] C.-H. Kuo and S.-C. Lin, "A delta-sigma modulator-based class-D amplifier," in *2016 IEEE 5th Global Conference on Consumer Electronics*, 2016, pp. 1–2.

[2] C.-H. Kuo and Y.-J. Liou, "A Delta-Sigma Modulator with UPWM Quantizer for Digital Audio Class-D Amplifier," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, 2019, pp. 293–297.

[3] J. L. A. de Melo, P. V. Leitão, J. Goes, and N. Paulino, "A simple class-D audio power amplifier using a passive CT $\Sigma\Delta$ modulator for medium quality sound systems," in *2015 22nd International Conference Mixed Design of Integrated Circuits Systems (MIXDES)*, 2015, pp. 543–546.

[4] M. Auer and T. Karaca, "A Class-D Amplifier with Digital PWM and Digital Loop-Filter using a Mixed-Signal Feedback Loop," in *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, 2019, pp. 153–156.

[5] N. Pereira and N. Paulino, *Design and Implementation of Sigma Delta Modulators ($\Sigma\Delta M$) for Class D Audio Amplifiers using Differential Pairs*. Cham: Springer, 2015.

[6] B. F. Bokmans, B. J. D. Vermulst, and J. M. Schellekens, "A Delta-Sigma Modulated Multi-MHz GaN Half-Bridge featuring Zero-Voltage Switching and Blanking Time Compensation," in *2021 IEEE 12th Energy Conversion Congress Exposition - Asia (ECCE-Asia)*, 2021, pp. 1638–1643.

[7] G. Meneghesso, M. Meneghini, and E. Zanoni, *Gallium Nitride-enabled High Frequency and High Efficiency Power Conversion*. Cham: Springer, 2018.

[8] M. Gwóźdź and D. Matecki, "Power electronics controlled voltage source based on modified sigma-delta modulator," in *2016 IEEE International Power Electronics and Motion Control Conference (PEMC)*, 2016, pp. 186–191.

[9] J. Daniels, W. Dehaene, M. Steyaert, and A. Wiesbauer, "A/D conversion using an Asynchronous Delta-Sigma Modulator and a time-to-digital converter," in *2008 IEEE International Symposium on Circuits and Systems*, 2008, pp. 1648–1651.

[10] F. Colodro, A. Torralba, and M. Laguna, "Continuous-Time Sigma–Delta Modulator With an Embedded Pulsewidth Modulation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 3, pp. 775–785, 2008.

[11] J. M. de la Rosa, "Sigma-Delta Modulators: Tutorial Overview, Design Guide, and State-of-the-Art Survey," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 1–21, 2011.

[12] M. Stork, "Asynchronous sigma-delta modulator and fast demodulator," in *2015 25th International Conference Radioelektronika (RADIOELEKTRONIKA)*, 2015, pp. 180–183.

[13] E. Roza, "Analog-to-digital conversion via duty-cycle modulation," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 11, pp. 907–914, 1997.

[14] T. Wolfer and E. Hennig, "An Active Feedback Coefficient Tuning Technique for Compensating Time-Constant Variations in Continuous-Time Delta-Sigma Modulators," in *2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS)*, 2021, pp. 17–20.

[15] Efficient Power Conversion Corporation, "EPC2152 80 V, 15 A ePower^{TM}, Stage Preliminary Datasheet, Revision 2.0," März 2021, [Stand 21. Juni 2022]. [Online]. Available: https://epc-co.com/epc/Portals/0/epc/documents/datasheets/EPC2152_datasheet.pdf.

Andreas Brunner erhielt den akademischen Grad B.Sc. in Mechatronik mit Schwerpunkt Elektrotechnik vom Management Center Innsbruck im Jahr 2020 und den M.Sc. in Leistungs- und Mikroelektronik von der Hochschule Reutlingen im Jahr 2022. Aktuell arbeitet er dort als wissenschaftlicher Mitarbeiter am Lehr- und Forschungszentrum Electronics & Drives im Bereich Leistungs- und Mikroelektronik.

Philipp Czerwenka erhielt den akademischen Grad B.Eng. von der Dualen Hochschule Baden-Württemberg Mannheim im Jahr 2020 und den M.Sc. an der Hochschule Reutlingen im Jahr 2022. Aktuell arbeitet er dort als wissenschaftlicher Mitarbeiter am Lehr- und Forschungszentrum Electronics & Drives im Bereich Leistungs- und Mikroelektronik.

Tobias Wolfer erhielt den akademischen Grad B.Eng. von der Dualen Hochschule Baden-Württemberg Stuttgart im Jahr 2016 und den M.Sc. an der Hochschule Reutlingen im Jahr 2019. Aktuell arbeitet er dort als wissenschaftlicher Mitarbeiter und Doktorand am Lehr- und Forschungszentrum Electronics & Drives im Bereich Modulatorforschung für die Leistungselektronik.

Eckhard Hennig erhielt den akademischen Grad Dipl.-Ing. von der Technischen Universität Braunschweig im Jahr 1994 und den Dr.-Ing. an der Universität Kaiserslautern im Jahr 2000. Er ist Professor für Integrierte Schaltungstechnik an der Hochschule Reutlingen. Seine Forschungsbereiche umfassen Low-Power-CMOS-Schaltungstechnik für Smart-Sensor-Anwendungen und Electronic Design Automation (EDA) für Analog/Mixed-Signal-Schaltungen.

The Effect of Image Coding on CMS Image Quality Parameters Using Embedded Video Coding of an MPSoC FPGA

Jannik Mehrke, Anestis Terzis, Oliver Bringmann

***Abstract*—One of the main sensors used to enable advanced driver assistance systems and autonomous driving are cameras. The image signal processing, required for this, is very demanding in terms of computing power and real-time requirements. The overall project validates the outsourcing of the very demanding image processing algorithm into an FPGA cloud. Therefore, a tradeoff between image quality of the compressed signal and real-time process is analyzed and discussed. When using more compression to the image, transmission of the data into the cloud is faster, but the quality of the image for the object detection is worse. This paper shows how the effect of image coding on image video parameter can be measured and validates the results.**

***Index Terms*—Camera Monitor System, Cloud, MPSoC FPGA, Hybrid Image Processing, Image Compression**

I. Introduction

Cloud-based data processing has become more important within the last decade. Scalability, distributivity, interoperability, modularity are some of the main advantages of cloud computing [1]. The consideration of new cloud-based architectures is a promising approach because of the increasing processing requirements for advanced automotive use cases [2]. A novel hybrid cloud-based processing platform, based on FPGAs, is designed and exemplary implemented within this work. The main purpose of the overall project is to generate a platform to enable investigations of the effects of different parameters on real-time cloud-based processing architectures. The chosen example architecture is the implementation of a Camera-Monitor-System (CMS) as Mirror replacement in vehicles as introduced in [3]. The normative framework for CMS as mirror replacement is defined in [4].

This paper focuses on the real-time image compression of the data that are transmitted to a cloud and on the effects of the compression on the image quality parameters. The purpose is to make the effect of image compression on image video parameter measurable. The overall system generates image data in a vehicle and transmits these data to a cloud system for object detection or scene interpretation. Because of the size of the generated raw-image data and the available bandwidth for the sending during driving, compression of the raw-data is indispensable. Therefore, two state of the art real-time embedded image coding algorithms are evaluated according to their processing time, compression ratio and effects on the image quality parameters with the introduced platform.

The effects on the image quality parameters are measured with a modified object detection algorithm. This algorithm is filled with generated videos. These videos are simulated night drives in the laboratory. The night scenario is chosen because of the low information content of the images . Vehicles are seen as point light sources (PLS) in night scenarios. The predominant part of the images is dark, so it does not include any information. Only some areas inside the images include information other than "darkness". Depending on the configuration of the image compression algorithms, they can plane those non-dark areas for example because of their minor size. With the planing of those areas, the information inside the image gets lost, which has major impacts on the object detection. For the validation of the effects, a night algorithm is used. This night algorithm detects point light sources in dark images. For the validation of the effects on image quality parameter, the results of detected vehicles before and after compression are compared. The results show major bias for the different compression algorithms with varying configurations.

This paper is organized as follows, the next section introduces the implemented and modified night algorithm. Section 3 focuses on the real-time image compression and the configuration possibilities. Section 4 defines the source material for the validation before section 5 shows and discusses the results of the measurements. At the end a conclusion is drawn.

II. Implementation of the Night Algorithm

The implemented MATLAB Night Algorithm is based on a "Haar algorithm" [5]. The Night algorithm uses Haar feature detection to detect point light sources (PLS) in the first stage. The second stage searches PLS with the same or close radius within a defined area around the source. When there is another PLS

Jannik Mehrke, Jannik.Mehrke@thu.de, Anestis Terzis, Anestis.Terzis@thu.de, both at University of Applied Science Ulm. Oliver Bringmann, Oliver.Bringmann@uni-tuebingen.de, Eberhard-Karls-University Tuebingen.

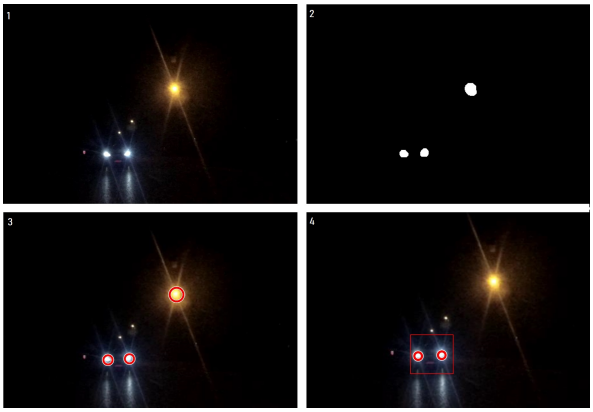


Figure 1. Steps of the Night Algorithm.

that matches the origin, those two PLS are merged to one detected vehicle. At the end, frames with detected vehicles and the number of vehicles are counted for the validation of the results. The PLS within this context all comply with the normative framework of the UN Regulation Number 46 [6].

In detail, the Night Algorithm works as follows: The binarization of the image is the first step to achieve a black and white image. Therefore, an offset value is defined to decide whether one pixel is black or white. Then a Haar-based approach is used for the edge detection as described in [5]. This approach detects transitions between black and white in the image. Once all the edges are detected, the MATLAB function "imfindcircles" [7] is used to detect circles along the found edges. This function provides their diameter and center position when a circle is detected. It searches circles in images whose radii are approximately equal to a radius within a defined range. The last step compares all found objects with each other. When two objects within a defined range have the same or similar radius, these two are merged to one vehicle. Figure 1 shows the individual steps of the night algorithm. The first image shows the raw data. The result of the binarization is visualized in the second image. In the third image, all detected circles are highlighted. This stage detects all lights in the image. This is the reason why the street light is also detected as a point light source. In the last picture, the final stage of the algorithm is visualized. Both point light sources of the vehicle are connected and highlighted.

The implemented Night Algorithm is not a state of the art benchmark algorithm. This algorithm was implemented to make the effect of compression measurable and for the validation of the concepts based on criteria of the UN Regulation Nr 46. This algorithm analyzes the generated video files offline after their generation. It uses elementary detection methods and is only suitable for the detection of normative point light sources. Within real driving scenarios there are various situations and constellations within the image for which this algorithm is not suitable.

Table I
RESOURCE UTILIZATION OF THE SENDER SIDE DESIGN.

Resource	Utilization	Available	Utilization [%]
LUT	58883	230400	25.56
LUTRAM	4774	101760	4.69
FF	81864	460800	17.77
BRAM	69	312	22.12
URAM	44	96	45.83
DSP	97	1728	5.61
IO	22	360	6.11
GT	3	20	15.00
BUFG	26	544	4.78
MMCM	3	8	37.50

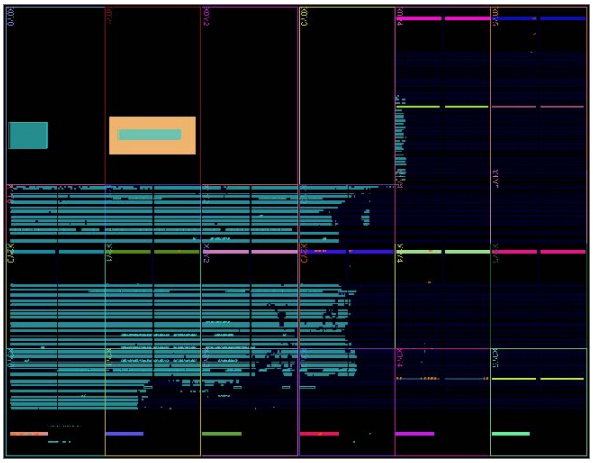


Figure 2. Resource usage on the chip of the sender side component.

III. IMAGE COMPRESSION

For the implementation of the introduced system, a Xilinx Zynq UltraScale+ MPSoC ZCU106 FPGA board [8] is used. The board provides many FPGA resources. The resource utilization is listed in Table I. Graphical representations of the resource utilization and the routing on the FPGA are provided in Figure 2 and Figure 3. The used SoC component is a state of the art FPGA-based SoC for modern ADAS and streaming applications. One main advantage of this chip is that it comes with a hardware implemented Video Codec Unit (VCU) [9] within the SoC. This VCU can be used to hardware-implement a H.264 or a H.265 Video Codec. Both are state of the art video codecs that are used at real-time streaming platforms. These two codecs come with a bunch of configuration options. The configuration features of primary focus are shortly introduced within the following sections. Another advantage of the FPGA-based system design is the energy consumption of the complete system. FPGA chips are more power efficient than CPU or GPU based chips according to a study of Inaccel [10].

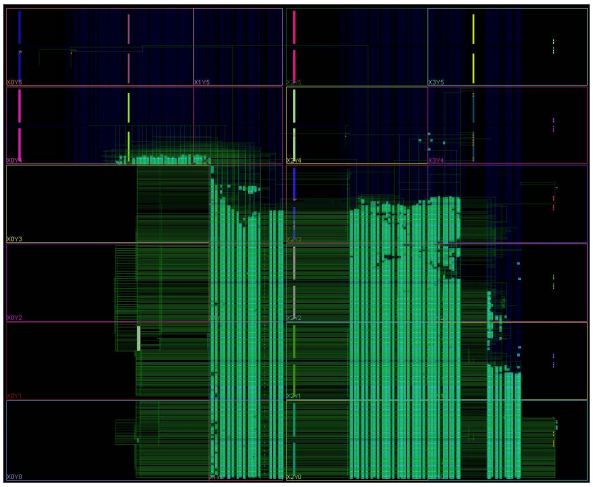


Figure 3. Routing on the chip of the sender side component.

The overall power consumption of the sending side design at full power is $6.195W$.

The relevant streaming pipeline for this paper consists of mainly three parts:

- the HDMI input
- the image pre-processing
- and the image compression and transmitting over Ethernet.

All parts are controlled or processed by the FPGA. The HDMI input provides the raw image data to the FPGA. The FPGA itself then performs the pre-processing. Within the pre-processing the image data can be scaled. The scaling includes a horizontal and vertical scaling of the image. Therefore, two factors are defined as scaling factors from the original size to a smaller size. Upscaling of the image is not implemented within this project, only downscaling is possible. In addition a frame rate splitter is implemented. With the splitter it is also possible to define a lower frame rate for the transferred data. The pre-processing block can be used as an additional data reduction block for further applications. With this block the image data can be pre-processed to reduce the amount of transported data. It is implemented as a black box that is running on the FPGA. The user sets the three parameters within the streaming pipeline to configure this block. By default all three parameters are set to 1. This causes that the raw image is transferred to the image compression. The last part is the compression and sending part. These are two hardware components that are only configured by the FPGA. The following sections describe the used configuration possibilities of the image compression in details.

A. Target bitrate

The variable target bitrate defines the allowed bit rate after the compression. The encoder will try to achieve the defined value to average throughout the video. The unit of target bit rate is $[\frac{kbit}{s}]$.

B. Number of Slices

The number of slices defines how many slices per frame are used with the H.265 encoder. The H.265 encoder is a so called differential encoder. It only sends differences of the actual frame to the previous frame. Therefore, it uses some kind of movement detection within the frame. This movement detection is done within slices. Each slice contains one or several full rows of the image in which differences are searched. The more slices are used, the more parallel operations can be done. The slices are spread over the frame as regularly as possible. A higher number of slices leads to a greater amount of required computational power. This is caused by a more parallel processing architecture. Each parallel process needs computational power to be processed as fast as possible. The number of slices that can be used on the SoC is limited by the processing power of the VCU. The hard-IP VCU component in the SoC is sufficient for parallel processing architectures until 16 slices. The calculation needs too many threads when using a higher number of slices which the processor can not process in parallel anymore. Then the system can not reach the real-time requirement because the overall processes take to long and each process is blocking the others by reserving and using the limited resources on the chip.

C. Latency Mode

The latency mode defines the used encoder latency mode, which defines the focus of the compression. There are two important modes for this system, the normal mode and the low latency mode. The focus of the normal mode is on quality while the focus of the low latency mode is on timing. Low latency mode can for example increase the number of slices for parallelism, which leads to less computational time but more computational power. It also provides subframe level output. This leads to the effect that the already processed parts of the images are forwarded while processing the other areas of the image. Common methods, like the normal mode, process the whole image before forwarding the results.

D. Configuration of the hardware components

For the running of the system, streaming pipelines need to be configured on the sending and receiving side. Therefore, videos are stored on the FPGA-based sending component. These videos are the basis for the compression and later for comparison of the results. The FPGA-based board is used to encode the videos and send them via UDP protocol over Ethernet to a computer. The receiving component decodes the UDP data and stores the decoded videos. On both devices the pipeline-based multimedia framework GStreamer [11] is installed and implemented to configure and start the streaming pipelines. Two different GStreamer

Table II
GENERATED VIDEOS FOR THE VALIDATION. LISTED ARE THE RESOLUTION (RES) [PX], THE FRAME RATE (FR) [FPS], THE VELOCITY OF THE VISUALIZED VEHICLES (VEL)[$\frac{m}{s}$], THE SIZE OF THE VIDEO FILES (SIZE) [MB] AND THE LENGTH OF THE VIDEO (LEN)[S].

Nr	Res[px]	FR[fps]	Vel [$\frac{m}{s}$]	Size[MB]	Len[s]
1	2560x1440	60	0,39	98,9	15
2	1920x1080	60	0,1545	95,6	35
3	1280x720	60	0,39	66,4	18
4	1280x720	60	0,1545	119,2	32
5	2704x1520	60	0,39	63,3	24
6	1024x576	15	N.A.	58,6	7

pipelines must be started on the different devices. Each pipeline configures the needed parameters for the different processing parts as described in the section III.

E. Precision and Recall

In the field of Object detection there are a variety of metrics available to compare different algorithm according to their accuracy. Two commonly used metrics are the Precision P and the Recall R. Algorithms sometimes detect objects without them really being in the image. The Precision represents those and provides a ratio. The Precision is defined as the number of correct detected objects divided by the total number of detected objects. This generates a number between 0 and 1. The higher the number, the better the algorithm. The second value is the Recall R. The Recall defines how many objects in the ground truth are detected correctly. Therefore, the Recall is defined as the number of correctly detected objects divided by the total number of objects in the ground truth. This also generates a number between 0 and 1. A higher number indicates a better result. Both variables are later used to evaluate the results of the detection. The number of detected objects in the raw images defines the ground truth. In case of a changed number after compression, an effect of the compression is measured.

IV. VIDEOS FOR THE INVESTIGATION OF THE ALGORITHM

For the measurement of the influences, different videos were produced containing different scenarios. For this paper, five laboratory videos and one real-driving video are used. The laboratory videos all show a model vehicle that drives with different constant velocities towards the camera. The camera is configured with different resolutions and fields of view for the recording. The real-driving video is a real-drive through a city at night. This is used to validate the theoretical results in real scenarios. The details of the videos are listed in the Table II.

Table III
BENCHMARK VALUES FOR THE DIFFERENT VIDEOS GENERATED WITH THE MATLAB NIGHT ALGORITHM.

Name	Frames with vehicles	Detected Vehicles
Video1	799	799
Video2	1348	1348
Video3	417	417
Video4	771	771
Video5	1278	1278
Video6	394	652

The different configurations for the cameras indicate different hardware components within the final mirror-replacement system. Because $Video6$ contains a real-driving scenario, the velocity of the pictured vehicles is unknown.

These videos are sufficient for the overall proof of concept. They represent a real driving scenario at night with different normative and reproducible configurations. The effects of different compression factors has major impacts on the quality of these videos and can be measured with the implemented night algorithm.

V. EFFECTS

This section presents the results of the investigation. While the coding time and compression ratio are directly measured, the effects of the compression on the image quality parameters are investigated with the night algorithm. For this investigation, the number of frames with detected vehicles in the uncompressed videos and the total number of detected vehicles in all frames are used as benchmark for the effects in the first step. The uncompressed videos showed the following results in the test (see Table III).

In the first step only the numbers of detected objects are compared. In the second step, the quality of the results is measured with the Precision and Recall functions. The results for all videos are nearly the same, therefore in the following sections only the results for $Video1$ are presented.

A. Compression ratio

The compression ratio of image coding is defined as the relative size of data representation produced by a data compression algorithm. To calculate the compression ratio for the different algorithms, the sizes of the videos before and after coding are compared. These rates are calculated for both H.264 and H.265 with different target bitrates. The higher the target bitrate, the less compression is needed to achieve this value. The results can be seen in Table IV.

These results are generated with the same configurations for both algorithms. The target bitrate values are configured from $50000\frac{kbit}{s}$ down to $1\frac{kbit}{s}$. These limits are chosen to have a wide spectrum of compression ratios from nearly uncompressed to extremely

Table IV
VIDEO SIZE FOR DIFFERENT TARGET BITRATES AFTER H.264 AND H.265 ENCODING.

Name	Size H.264 enc. [MB]	Size H.265 enc. [MB]
Raw	98,9	98,9
TB 50000	76,9	76,9
TB 40000	61,5	61,5
TB 30000	64,2	64,1
TB 20000	31,1	30,8
TB 10000	15,800	15,600
TB 7500	12,000	11,700
TB 5000	8,060	7,980
TB 4500	7,270	7,000
TB 4000	6,510	6,230
TB 3500	5,660	5,450
TB 3000	4,840	4,630
TB 2500	4,040	3,860
TB 2000	3,170	3,100
TB 1500	2,390	2,350
TB 1000	1,610	1,590
TB 750	1,230	1,220
TB 500	0,850	0,864
TB 250	0,520	0,483
TB 100	0,517	0,363
TB 50	0,478	0,369
TB 1	0,505	0,384

compressed. While $50000\frac{kbit}{s}$ is achievable for the Video Codec, $1\frac{kbit}{s}$ could not be achieved. The Video Codec adjusted itself to a value close to the configured one. That is the reason why the results for $1\frac{kbit}{s}$ and $50\frac{kbit}{s}$ are incorrect and not representable results. The first correct result is at $100\frac{kbit}{s}$. This investigation shows that the H.265 algorithm is slightly better than the H.264 algorithm according to the achieved data size after compression. This directly causes a faster transmission of the data to the cloud. The results are as expected at the beginning of this investigation. The next step is to discuss the effects of the target bitrate on the object detection.

B. Effects of target bitrate

The night algorithm can directly measures the effects of the target bitrate. Therefore, the video-files with different target bitrate configurations are loaded into the Night algorithm. In first instance the algorithm only counts the frames with detected vehicles. The next step is to have a closer look at the results and calculate their Precision and Recall. The results of the frames with detected vehicles can be seen in Figure 4. The results for the Precision for different target bitrates as shown in Figure 5.

The results show that right from the beginning the H.265 algorithm is better than the H.264 algorithm. While decreasing the target bitrate, the quality of the video also decreases. At some point, the algorithms can not detect the point light sources correctly any more. Then in some frames, more than one PLS is detected at the real position (false-positive detection) or sometimes

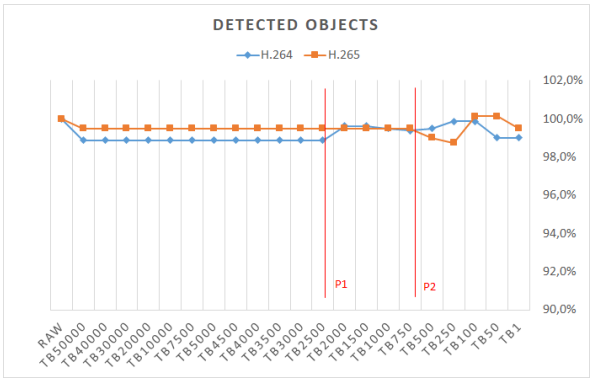


Figure 4. Percentage of frames with detected vehicles compared to the raw video for different target bitrates.

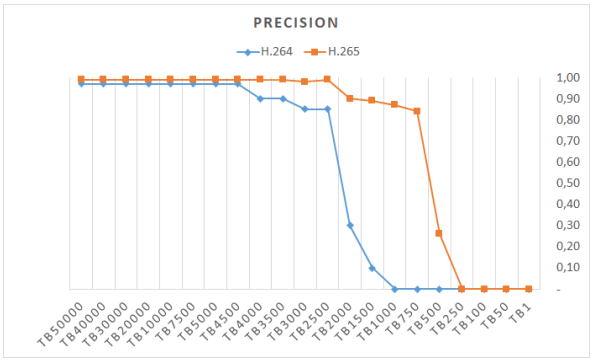


Figure 5. Precision for the codecs for different target bitrates.

none is detected (false-negative detection). This leads to an incorrect result and causes a bad Precision and Recall. Within these graphs two lines are drawn. These two lines indicate the points where Precision and Recall got bad for the compression algorithm. At point $P1$ the H.264 algorithm leads to bad results. At this point the number of false-positive and false-negative detections increases drastically. With a target bitrate of $2500\frac{kbit}{s}$ and lower the H.264 algorithm is not sufficient for the task anymore. The H.265 algorithm can achieve significant better results until point $P2$. With an target bitrate lower or equal to $750\frac{kbit}{s}$, the H.265 algorithm is not sufficient anymore. This result is also as expected, while H.264 sends the whole frame, H.265 only sends the differences between the last frame and the current frame. Therefore, the amount of data to send is reduced and so it was expected to achieve a higher compression without quality loss. At the part after point $P2$ the lines begin to vary. This is due to the fact that the quality of the image is too bad such that many objects are detected although there is only one PLS. The Precision and Recall at this stage become bad.

C. Effects of number of slices

The effects of the number of slices is measured the same way as the effects of the target bitrate. The

Table V
AVERAGE TIMING FOR THE H.264 AND THE H.265 ALGORITHM FOR DIFFERENT PROFILES.

Name	Encoding time [s]	Decoding time [s]
H.265	7.3940	9.9288
H.264	7.2215	10.1485

number of slices are variate for a constant target bitrate. The major effect is seen in the needed bandwidth for the transportation. The number of slices has neither significant impact on the quality of the image nor on the time needed to encode or decode the image. The major impact is on the transportation. Each slice calculates its area and sends the results when finished. When one slice is used, the whole image is processed by this slice and at the end the whole image is send. When 12 slices are used, then each slice is processing its area and sends its results when finished. This generates 12 packets of frames that are sent compared to only one packet. For the transportation this means that less bandwidth at the same time is needed because of the separation of the packet caused by different processing times of the slices. As mentioned before, the overall time is nearly the same in all configurations because this depends mainly on the target bitrate.

D. Effects on timing

The results of the timing are presented in Table V. This table shows the average timing over all profiles for the H.264 and H.265 coding. It shows that in average the H.265 encoding takes more time than the H.264 encoding. This is due to the fact of the higher complexity of the H.265 encoding. As mentioned before, H.265 detects the differences within two frames and only encodes those parts while H.264 is encoding the whole image with the same scheme. For the decoding, the H.265 algorithm is much faster than the H.264 decoding. This is due to the low data size that is needed to be decoded. This validation does not include the transmission of data to the cloud. As mentioned in section Table IV the size of the H.265 video is smaller than the H.264 encoded videos. In total the H.265 algorithm leads to a much faster transmission of the data although the computational complexity of the encoding is much higher.

E. Results

The investigations shows that the H.265 coding is more sufficient for real-time image compression than the H.264 coding. The faster process- and transportation time and the higher achievable compression cause this. The results are as expected, the new part within this paper is that now it is possible to measure the effect of the compression. From now on it is possible to define the lower boundary of the data transmission in the overall system concept. It was known that a higher compression generates a lower quality of the image but with lower coding- and transportation-latency. As mentioned in the beginning of this paper, the overall project investigates a real-time cloud processing architecture. Depending on the quality of the connection between the vehicle and the cloud, the image compression can now be adjusted. For low-bandwidth connections, the target bitrate can be reduced to guarantee a working system. When the bandwidth is too low such that a target bitrate lower than $750\frac{kbit}{s}$ is needed for the H.265 algorithm, the system must shut itself down due to bad results caused by bad quality of the images in night scenarios.

VI. CONCLUSION

In this paper, the effects of image compression on image quality parameter are measured and discussed. Therefore, an object detection algorithm for normative point light sources (PLS) in night scenarios was implemented. This algorithm makes the effect of image compression for a specific CMS application measurable by counting detected vehicles and calculating the Precision and Recall. Several videos are generated with different configurations for different codecs. These videos are analyzed with the implemented algorithm and the results are compared. The evaluation platform is implemented on an FPGA-based SoC board. The effect on the image quality mainly depends on the compression ratio of the used video codec. A lower target bitrate leads to a higher compression but reduces the quality of the image. This reduction can be seen in the results by fewer detected vehicles and a worse Precision and Recall. As expected at the beginning, the H.265 video codec is more sufficient for the real-time image compression than the H.264 algorithm. The main advantage of this research is that now it is possible to find a minimum configuration for the compression with different codecs. When for example using a H.265 codec at night scenarios, it is possible to reduce the target bitrate until $750\frac{kbit}{s}$ without major impact on the Precision of the system. In the overall system, the configuration of the video codec is adjustable depending on the available cloud connection parameters. For example with more bandwidth available for the transmission of data into a cloud, a lower compression is used to achieve better detection results. The research in this paper generated a lower boundary for the transmission in night scenarios. When the available bandwidth of the cloud connection is below $750\frac{kbit}{s}$ for the H.265 algorithm, it is not sufficient to use the object detection in the cloud. In this case the system must be shut down until a "better" connection is available. Otherwise the amount of false-positive or false-negative results increases caused by the low image quality. In a future work, this investigation is done for day scenarios.

REFERENCES

[1] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–10.

[2] A. Iordache, G. Pierre, P. Sanders, J. G. D. F. Coutinho, and M. Stillwell, "High performance in the cloud with fpga groups," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Dec 2016, pp. 1–10.

[3] A. Terzis, Ed., *Handbook of Camera Monitor Systems - The Automotive Mirror-Replacement Technology based on ISO 16505*. Springer International Publishing, 2016, vol. 1, no. 10.1007/978-3-319-29611-1.

[4] "Road vehicles — ergonomic and performance aspects of camera monitor systems — requirements and test procedures," *International Organization for Standardization*, 2019.

[5] D. Peleshko and K. Soroka, "Research of usage of haar-like features and adaboost algorithm in viola-jones method of object detection," in *2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, 2013, pp. 284–286.

[6] "Regulation No 46 of the Economic Commission for Europe of the United Nations (UNECE) - Uniform provisions concerning the approval of devices for indirect vision and of motor vehicles with regard to the installation of these devices," Addendum 45, Revision 6, 2016.

[7] MATLAB. (2020, Sep.) imfindcircles describtion. [Online]. Available: https://de.mathworks.com/help/images/ref/imfindcircles.html

[8] Xilinx, *ZCU106 Evaluation Board*, 2019.

[9] X. , *H.264/H.265 Video CodecUnit v1.2 - LogiCORE IP Product Guide*, 2020.

[10] Inaccel, "www.inaccel.com," *Inaccel*, February 2022.

[11] GStremer, *GStreamer - API reference guide*, 2020.

Jannik Mehrke studied electrical engineering and Information technology at the Ulm University of Applied Science. During his Bachelor study, he specialized on automotive electronics. Later he finished his Master's degree at the Technical University of Munich where he also studied in the field of electrical engineering with specialization on automation technologies. Prior to this he worked two years in an engineering office in the field of automated testing of vehicles. Since 2018, Jannik Mehrke works as a researcher at the Ulm Technical University of applied science in the field of embedded systems. His research focus on cloud-based hybrid image processing architectures on SoC platform.

Anestis Terzis is professor for digital systems design and the head of the Institute of Communication Technology at Ulm University of Applied Sciences in Germany. Prior to this, he was with Daimler AG for ten years and worked in the Group Research and Mercedes-Benz Cars Development, with a main focus on future advanced driver assistance systems. Anestis Terzis received his diploma in Communications Engineering from Ulm University of Applied Sciences and his doctoral degree in Electrical Engineering from the University of Erlangen-Nuremberg. He is an expert member of the standardization and regulation committees (ISO, SAE, IEEE) in the field of camera monitor systems.

Oliver Bringmann received the Diploma (M.S.) degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, and the Doctoral (Ph.D.) degree in computer science from the University of Tübingen, Germany, in 2001. He was with the FZI Research Center for Information Technology, Karlsruhe, in various positions as a Department and Division Manager and a Member of the management board, until 2012. He has been a Professor and the Director of the Chair for Embedded Systems with the University of Tübingen since 2012, where he is also serving as the Vice Head of the Department of Computer Science since 2014. He has authored and coauthored of more than 220 publications in the area of electronic design automation, embedded system design, and SoC architectures for automotive electronics and edge devices. His current research interests include electronic design automation, embedded system design, timing and power analysis of embedded software, embedded AI architectures, hardware-enhanced security, and robust perception. Oliver Bringmann is an IEEE member.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN
Hochschule Albstadt-Sigmaringen
Albstadt-Sigmaringen University

Inertial Navigation – Importance of Initial Heading: A Constrained Dual GNSS Receiver Approach

Lukas Blocher, Tobias Hiller, Wolfram Mayer, Joachim Gerlach, Oliver Bringmann

***Abstract*—This paper describes scenarios for automated driving which are highly relevant on accurate initial heading estimation. We show thoughtfully designed experiments which are closely aligned with real world demands to explain the relevance and influence of initial heading estimation. We conclude that purely inertial navigation is highly dependent on accurate initial heading. On the basis of these experiments we show a constrained dual GNSS receiver approach which takes advantage of short baselines and known receiver baseline length to estimate initial heading without relying on the Earth's magnetic field.**

***Index Terms*—IMU, inertial navigation, GNSS, GPS, initial heading, compass, dual receiver, constrained**

I. Introduction

Current developments in automated driving show a continuous extension of the operation design domain to lower and higher speeds as well as more challenging environments. Since an automated vehicle is basically a mobile robot, both applications share the common fundamental task of navigation. *Navigation* is the job of making a plan from a start to a destination and therefore requires determining the current position, a process called *localization* [7, p.241]. This work deals with the importance of one special component of localization: *Initial heading*. Localization is the basic building block of every mobile robot [8, p.385], therefore the heading component deserves special attention to provide a sophisticated foundation for further enhancements in automated driving and high level decision making.

A. Initial Heading

Heading is part of an attitude solution determining the rotation with respect to a given coordinate frame. Attitude includes three rotation angles: yaw/heading, pitch and roll [6, p.32]. While pitch and roll can be estimated by using the gravity vector, heading ψ requires special treatment since external reference and separate sensors are required to determine heading relative to north. Such references may be a compass or a GNSS receiver. A compass may be challenged by magnetic field fluctuations caused by electric drives whereas a single GNSS receiver may only deliver heading when using a trajectory at least two consecutive, sufficiently spaced, samples. To address these challenges this work relies on a dual GNSS receiver approach to be able to determine heading without any magnetic reference in low speed and standstill situations. In this context the term *initial* denotes two special scenarios which are highly relevant when *real* autonomy of robots is desired:

1) System Cold Start: Start of a mobile robot without any prior knowledge of heading while being at standstill.

2) Purely Inertial Navigation Start: The robot is moving along a trajectory with known heading. At the described point in time continuous heading determination is becoming impossible due to the GNSS reference system being unavailable. The last known heading needs to be extrapolated by using inertial sensors.

Lukas Blocher, Tobias Hiller, Wolfram Mayer, Robert Bosch GmbH. Joachim Gerlach, gerlach@hs-albsig.de, Albstadt-Sigmaringen University. Oliver Bringmann, oliver.bringmann@uni-tuebingen.de, University of Tuebingen.

B. Inertial Navigation

Inertial sensors are self-contained and therefore not dependant on any external source. Consequently this type of sensor does require an exact knowledge of initial position and attitude at start when used for a navigation task. Without having a global reference the knowledge of position relative to a starting point is a challenging task [7, p.242]. An extrapolation of position is done by integrating angular rate from three gyroscopes and acceleration from three accelerometers relative to a starting point. Special attention is brought to inertial navigation when a robot enters a GNSS denied environment or faces system degradation and therefore loses its absolute positioning reference. The challenge is to initialize an inertial navigation system properly, especially in the initial heading scenarios listed above.

C. Approach

The following sections explains the relevance of initial heading and the defined scenarios by first focusing on purely inertial navigation. The experiments clearly show the influence and importance of a reliable and

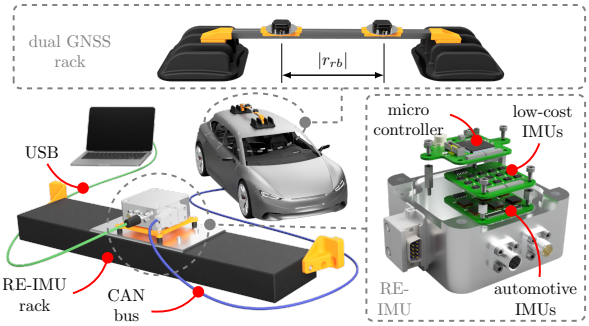


Figure 1. [3] ©IEEE 2022. Rendering of the setup for our experiments built around the redundant IMU (RE-IMU), which is mounted on a rack in the trunk of a vehicle. The RE-IMU is connected to vehicle CAN bus to acquire wheel speed information. *Detail, right*: exploded view of the RE-IMU showing its sub-components. *Detail, top*: Dual GNSS antenna rack for ground truth and heading acquisition.

Table I
TOTAL MEAN OF MEASURED GYROSCOPE ANGLE RANDOM WALK (ARW) AND BIAS INSTABILITY (BIS) FOR ALL SENSORS COMBINED IN THE RE-IMU AS EXPLAINED IN [3][4].

Parameter	x**-axis**	y**-axis**	z**-axis**	**Unit**
ARW	2.06	2.36	1.54	[mdps/rtHz]
BIS	0.40	0.68	0.46	[dph]

accurate heading information for the previously defined scenarios. On the basis of the conclusion in II. the concept of the dual GNSS receiver approach is explained to address the previously discovered challenges for initial heading estimation.

II. Importance of Initial Heading

This first section focuses on inertial navigation and explains the relevance of heading in this context. As concluded in our previous work purely inertial navigation with low-cost MEMS IMUs is only relevant for real-world applications for short periods of time. Therefore we introduced augmentation with vehicle odometry from wheel speed sensors. In our previous work purely inertial navigation and addressing stochastic and deterministic errors with different sensor array configurations is explained more detailed in [2] [3]. This work summarizes the previous conclusions and reconsiders them within the use-case of automated driving. The approach of this section is called gyroscope-assisted odometry (GAO) which receives velocity information from wheel-speed sensors while calculating relative heading from angular rates. In this case one integration step can be removed compared to purely inertial navigation and the duration of usable navigation can be extended.

A. Setup

The experiment setup is built around the redundant IMU (RE-IMU). It has to be stated that for this work

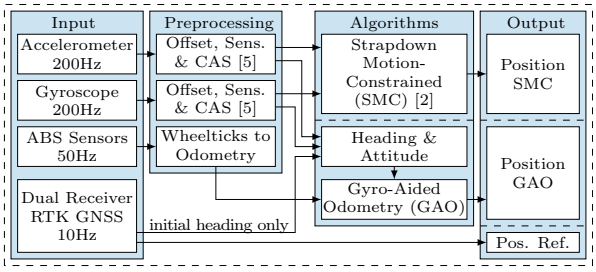


Figure 2. [3] ©IEEE 2022. Block diagram of data-processing. Pre-measurement calibration is executed to reset gyroscope offset, normalize accelerometer data and to calculate attitude. Vehicle wheel ticks are converted to distance and velocity. GNSS is used for reference trajectory and initial heading initialisation during calibration.

the exact sensor configuration within the IMU is of minor importance, more information may be obtained from [3]. The RE-IMU is also responsible for CAN data acquisition and data streaming to a laptop. The IMU is rigidly mounted to the trunk of the vehicle while sitting on vibration dampers, as shown in Fig. 1. The micro controller within the RE-IMU handles sensor SPI data communication, data streaming via USB and also collecting wheel tick information from the ESP system. We took care to allow high speed and precisely synchronized 50Hz wheel tick data acquisition. Additionally two multi-band GNSS antennae with a baseline length of 32 cm are mounted to the roof of the vehicle. Two uBlox F9P GNSS receivers are linked in moving baseline configurations and therefore are able to provide heading information. Additionally the receiver positions are corrected by SAPOS HEPS real time kinematic service. The usage of RTK improves absolute positioning for allowing precise ground truth acquisition. While RTK improves absolute positioning, heading is not very sensitive to errors in position [9]. The GNSS setup allows sub-10 cm absolute positioning and sub-0.5 ° heading acquisition at high rate.

B. Theory

1) Inertial Sensors: The output of the RE-IMU can be treated as one high performance MEMS IMU due to taking the mean of all 14 sensor samples resulting in enhanced noise performance. The detailed architecture and IMU data fusion is described in [2]. Additional sensor array architectures were analysed in [11]. The sensor fusion of the individual IMUs result in angle random walk (ARW) and bias instability (BIS) parameters for RE-IMU total output as shown in Tab. I.

2) Virtual Wheel Model: The setup uses the rear axle as centre of reference for odometry calculation as discussed by [3] and [10]. The risk of introducing additional wheel slip is reduced by extracting odometry data from the rear axle while using a front wheel drive vehicle. Both rear wheels are considered as one virtual middle wheel in the reference point, while assuming constant velocity between samples. Relative motion

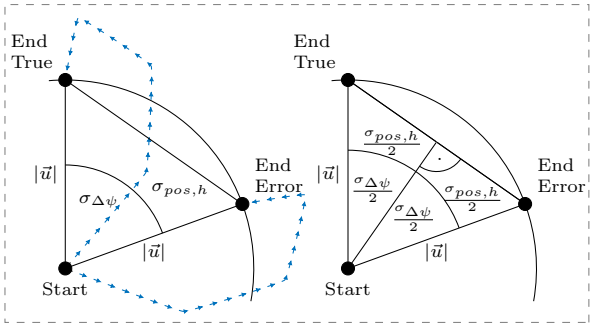


Figure 3. Illustration of the influence of initial heading on the scenario as shown in Figure 4. The initial heading error $\Delta\psi$ and the distance between start and endpoint $|\vec{u}|$ directly influence position error on the endpoint.

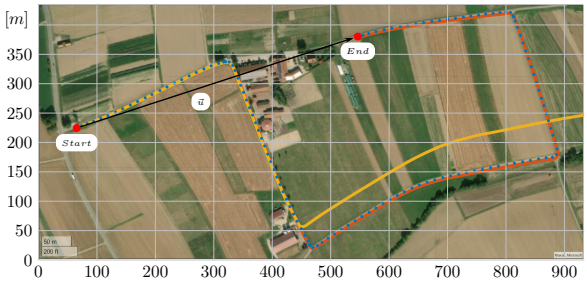


Figure 4. [3] ©IEEE 2022. Comparison of output trajectory of experiment D, shown in Figure 5. *Blue dots:* RTK-corrected GNSS ground truth, *yellow*: motion constrained strap-down, *red:* gyro-aided odometry. The scenario involved a distance travelled of 1600 m between start and end point (*red*) in 275 s.

heading is provided integrating gyroscope angular rate output.

3) Initial heading and Attitude: Initial heading and attitude are fundamental components to be able to make use of the odometry data. A coordinate transformation is required to transform odometry data from the body frame to the navigation frame based on current attitude. The discussed signal flow is shown in Fig. 2. To determine the initial attitude, the gravity vector is derived from accelerometer data. Using the initial attitude as a starting point continuous attitude is maintained by integrating angular rates. As illustrated in Fig. 3 the initial heading error $\Delta\psi$ together with the distance between start and endpoint $|\vec{u}|$ directly influence position error on the end point. The initial heading position error component $\sigma_{\text{pos},h}$ is time invariant and given by

$$\sigma_{\text{pos},h}(|\vec{u}|) = 2|\vec{u}|\sin(\frac{\sigma_{\Delta\psi}}{2}). \quad (1)$$

The error contribution of z-axis gyroscope $\sigma_{\text{pos},g}$ resulting from ARW N_z and BIS B_z is dependent on both time t and average vehicle velocity $\bar{v}$

$$\sigma_{\text{pos},g}(t,\bar{v}) = \bar{v}\sqrt{(2N_z\sqrt{t^3}/3)^2 + (B_z t^2/2)^2}. \quad (2)$$

Since both components $\sigma_{\text{pos},h}$ and $\sigma_{\text{pos},g}$ are stochastically independent the resulting total position error is defined as

$$\sigma_{\text{pos}}(t,\bar{v},|\vec{u}|) = \sqrt{\sigma_{\text{pos},g}(\bar{v},t)^2 + \sigma_{\text{pos},h}(|\vec{u}|)^2}. \quad (3)$$

4) Navigation Algorithms: The data processing in MATLAB is designed as shown in Fig. 2. After sensor data preprocessing and scaling, calibration data and alignment procedures are applied. The mean initial heading from the dual GNSS setup is used for absolute heading alignment during calibration. Applying the initial heading as offset to integrated angular rates results in converting the otherwise relative information to absolute north heading. During vehicle movement the GNSS heading is solely used for generating a reference trajectory for performance evaluation. Additionally the GNSS samples on the starting point are used to align the relative navigation information to a global map for visualisation.

C. Experiments

1) Calibration: In advance of the experiments the individual IMUs are calibrated using a highly accurate rate table. The first step of data processing applies calibration results to trim scale-factor, offset and cross-axis sensitivity. To achieve high positioning accuracy calibration is inevitable. Additionally the wheel circumference was calibrated using the RTK corrected GNSS setup to determine the distance per wheel tick. Preparatory to each experiment the vehicle standstill is used to determine gyroscope offsets, to calculate pitch/roll angles, to normalize the gravity vector and also to extract initial heading via the dual-antenna GNSS system.

2) Measurements: The measurements A to E, shown in Fig. 5 were recorded with the trajectory depicted in Fig. 4. The driven path is 1600 m long at an average speed of $5.8\,\text{m}\,\text{s}^{-1}$ within 275 s. To generate the reference trajectory one F9P GNSS chipset was configured in dual band and multi-constellation mode as well as fed with network RTK data. An open sky scenario, free from buildings and obstructions was chosen to ensure optimal GNSS signalling conditions and low risk of multipath.

3) Discussion: The values of z-axis in Tab. I being $N_z = 1.54\,\text{mdps/rtHz}$ and $B_z = 0.46/\sqrt{2\cdot\ln(2)/\pi}\,\text{dph} = 0.69\,\text{dph}$ are used for further predictions. With $t = 275\,\text{s}$ and $d = 1600\,\text{m}$ the corresponding mean velocity is $\bar{v} = 5.8\,\text{m/s}$. Using Eq. (2) the predicted gyroscope position error is $\sigma_{\text{pos},g} = 0.88\,\text{m}$. Since we did not include any reference for initial heading, we assume an error $\sigma_{\Delta\psi} = 0.4^\circ$ due to observations during our experiments. With $|\vec{u}| = 505.41\,\text{m}$ the calculated heading error is $\sigma_{\text{pos},h} = 3.53\,\text{m}$. The total position error is $\sigma_{\text{pos}} = 3.64\,\text{m}$, when using Eq. (3). The 2D position error $|\Delta\vec{d}| = |[x_n\ y_n]|$ between SMC, GAO

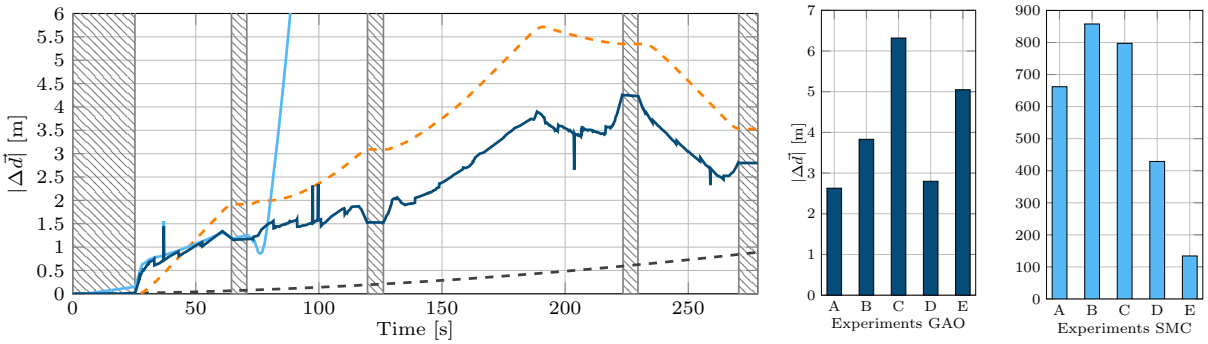


Figure 5. [3] ©IEEE 2022, *Left:* 2D position error plot of SMC (*solid, light blue*) and GAO (*solid, dark blue*) compared to the RTK GNSS reference corresponding to experiment D. The plot results from the trajectory shown in Figure 4. Stops at intersections are represented by the hatched grey areas. The predicted position error $\sigma_{\text{pos},h}$ (*dashed, orange*) based on the initial heading error decreases when returning closer to the start point. Predicted position error due to gyroscope noise $\sigma_{\text{pos},g}$ (dashed, grey) grows over time. *Right:* Total error of 5 independent measurements. All trajectories were as shown in Figure 4.

and GNSS reference is shown in Fig. 5. The SMC approach reflects the trajectory pretty well but shows significant error growth after a short period of time. The GAO approach does not suffer from cubic error growth due to the initial heading error being directly dependent of distance between start and end point $|\vec{u}|$. For $t > 230\,\text{s}$ the error decreases since $|\vec{u}|$ shrinks when closing in on the starting point.

D. Conclusion

Compared to a purely inertial approach, the odometry based variant shows greatly improved performance by being independent of accelerometer based velocity. High data rate and precisely synchronized data processing combined with dual-antenna GNSS baseline estimation provide an essential foundation for better navigation results. The measurements try to simulate the scenario "*System Cold Start*", defined in Sec. I. By predicting the individual error components we are able to highlight the relevance of initial heading estimation for such a scenario. For our experiment durations initial heading error is dominant over gyroscope error. Additionally we conclude that when not relying on the Earth's magnetic field, a state of the art dual-antenna GNSS system is suitable to solve for initial heading at standstill. Our observations fully match previous works like [7, P. 244] stating "*Especially bad are errors in orientation, because they have the largest effect on position accuracy*".

III. Constrained Dual GNSS Receiver Approach

Based on the resulting conclusions, the next step is to implement a solution for determining initial heading. Additionally such an approach can be merged with purely inertial navigation to allow continuous heading estimation. To be able to determine heading when at standstill for the scenarios described in the introduction we chose to implement a dual GPS approach with focus on the following criteria: I) short baseline $< 1\,\text{m}$,

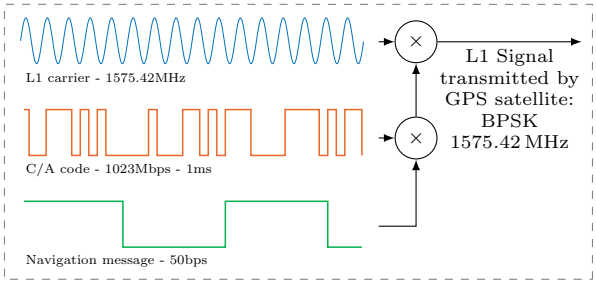


Figure 6. Data is applied to the carrier using BPSK (biphase shift key modulation). Carrier, C/A code and navigation message are combined to generate DSSS (direct spread spectrum) signals within the satellite (not drawn to scale) [13, 9].

II) double differencing for robust relative positioning and III) using the known antenna baseline length as a constraint. The setup consists of two independent GNSS receivers, receiving GPS L1 signals, while being clocked from an individual clock. The corresponding antennae are mounted on a antenna rack fixed to each other, as shown in Fig. 1. This work uses GPS to verify the developed approach which may be extended to other satellite constellations.

A. Fundamentals

GPS offers several information sources which are relevant for positioning. The basic one is the carrier wave. Measuring the carrier phase introduces most ambiguity but offers the most precise measurements. Since we are targeting for small baseline the approach shown makes use of the carrier wave, instead of the C/A code, due to the receiver noise being approximately $1\,\%$ of the wavelength [14, P. 53]. On top of the carrier wave the C/A code is applied to distinguish the satellites in the constellation. Therefore each satellite broadcasts its individual PRN (pseudo random noise) code. The satellites also provide navigation messages which contain ephemeris data for localizing the satellites in orbit. For this work the ephemeris data are required to determine the satellite line of sight vectors

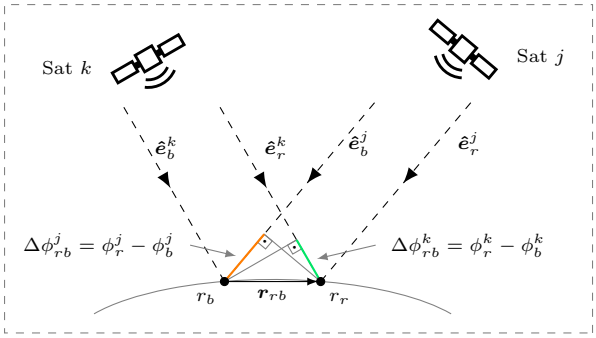


Figure 7. Two single carrier phase differences, as described by [12],[14, p.56]. Double differencing results from subtracting two single differences.

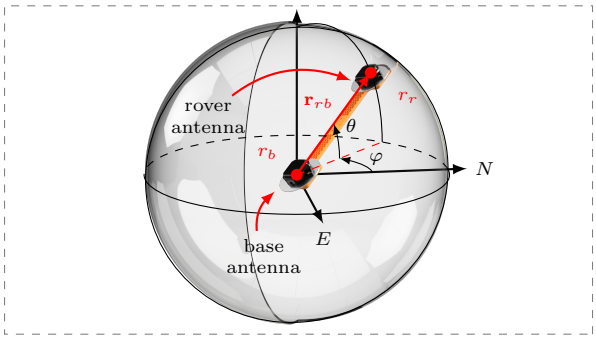


Figure 8. Spherical solution space with rover antenna in the centre of the sphere and radius being the antenna distance or baseline length $\boldsymbol{r}_{rb}$. The variables φ and θ are estimated, all possible solution candidates are located on the surface of the sphere.

$\hat{e}_b^k$. Instead of relying on absolute positioning we rely on using the advantage of having two receivers. Therefore relative positioning is possible using a technique called *"double differencing"*, an explanation in [14, p.58]. The single difference involves observing a satellite from both receivers which eliminates satellite clock errors and signal propagation delays due to atmospheric disturbances. Single differencing is executed by taking the carrier phase measurements of each receiver concerning the same satellite at the same point in time and subtracting them, as shown in Fig. 7. The step to double differencing is done by subtracting two single differences from two satellites k and j, consequently removing receiver clock errors if the receivers are synchronized. Double differencing removes hard to determine error sources leading to a very robust but relative output.

B. Solution Space

Most of the previous approaches try to estimate the baseline vector r_{rb} with three degrees of freedom X, Y and Z in a Cartesian coordinate system. This approach uses the known baseline length $|r_{rb}|$ as a constraint, since it is a-priori knowledge and decreases the size of the solution space. The resulting solution space is a sphere with radius $|r_{rb}|$. The base antenna is fixed

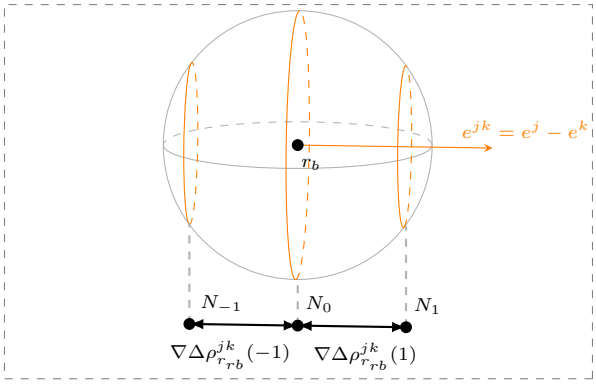


Figure 9. Positioning of the solution candidate circles for the double differenced LOS vector e^{jk}, when iterating N to solve for the unknown wavelength ambiguity.

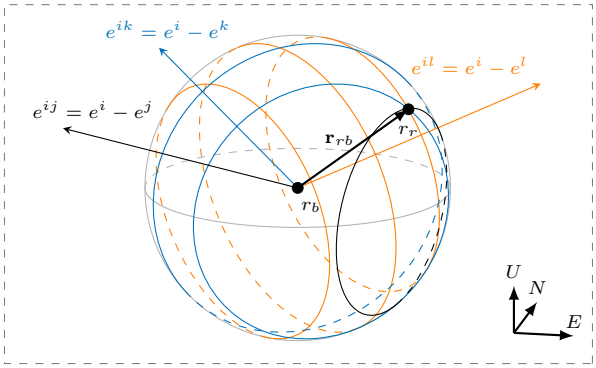


Figure 10. Solution space for 3 pairs of satellites, as shown in Fig. 9. The colors visualize all candidates for one pair of satellites when iterating N. r_r marks the point on the sphere surface where one circle of each satellite pair intersects and therefore the desired solution candidate.

in the centre of the sphere while the rover antenna can move on the surface of the sphere. Consequently we are searching for possible solutions in a spherical coordinate system with the two degrees of freedom φ and θ, as shown in Fig. 8.

The carrier phase measurements ϕ are geometrically aligned along the direction of the satellite signal represented by the line of sight (LOS) unit vectors $\hat{e}$. So double differencing can be considered creating a "virtual" satellite with the LOS vector $\hat{e}^j - \hat{e}^k$ and the geometric double difference $\nabla\Delta\rho_{r_{rb}}^{jk}$ being aligned along it. Due to an ambiguous carrier phase considering multiples N of the wavelength λ is required. As shown in Fig. 9 this results in multiple circles for each N with the circles being spaced by λ, scaled with $\hat{e}^j - \hat{e}^k$:

$$\nabla\Delta\rho_{r_{rb}}^{jk}(N) = \lambda\frac{(\nabla\Delta\phi_{r_{rb}}^{jk} + \nabla\Delta N_{r_{rb}}^{jk})}{|\hat{e}^j - \hat{e}^k|}. \qquad (4)$$

Each virtual satellite generates a group of circles for varying N, represented by colors in Fig. 9. The solution candidate we are looking for is a point on the sphere surface where one circle of each virtual satellite

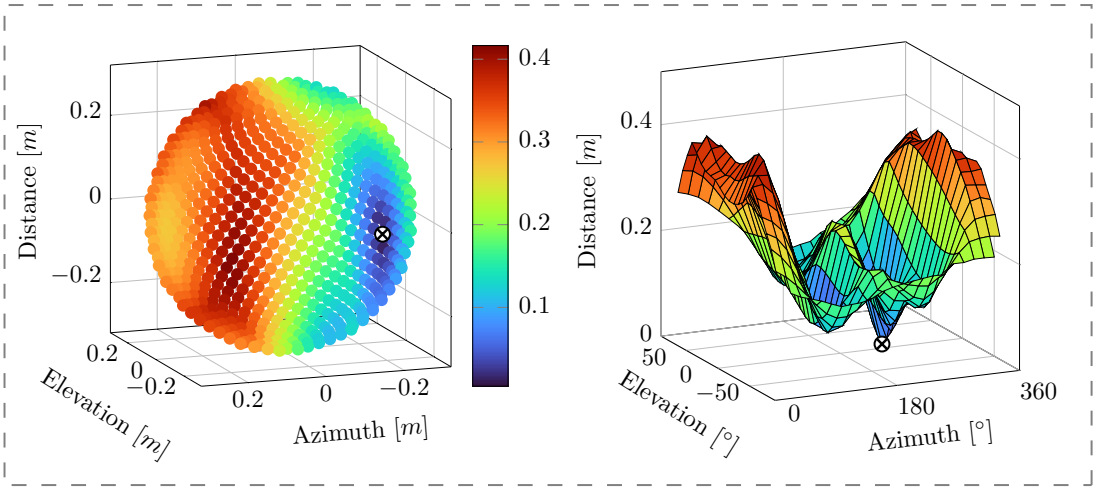


Figure 11. Discretised solution space with points projected onto the sphere surface. For each point the distance to the closest candidate circle, as shown in Fig. 10, is calculated. The resulting metric is the norm distance to each of the circles accumulated. *Left:* Spherical solution space, distance metric visualized by color. *Right:* Equivalent solution space as *left*, but azimuth and elevation angles rolled out in a plane and distance metric visualized on the z-Axis to visualize the distribution of minima.

intersects. The intersection point is the position of the rover antenna r_r relative to the base.

C. Simulation

For verifying the described approach, we developed a simulation environment to be able to freeze the highly dynamic satellite constellation and to isolate error sources. For finding a solution for a given scenario and satellite placement we set the baseline length to $|\boldsymbol{r}_{rb}| = 32\,\mathrm{cm}$ and projected a grid of points onto the sphere surface. For each point on the surface we calculated the distance to the closest solution circle to find possible intersection. The point being on or close to a circle of each group is the desired solution. The described approach is visualized in Fig. 9. The colors represent the norm distance to the closest circle of each group. Consequently the lower the metric, the better the potential solution. The dark blue points represent the desired solutions with a low distance metric. Fig. 11 *right* is just another representation of the solution space with azimuth and elevation angles φ, θ being rolled out in a plane. The Z-Axis represents the distance metric and therefore the distribution of minima is more obvious.

D. Conclusion

The simulation results are very promising and allow to evaluate different geometric constellations of satellites. The amount and placement of satellites have shown to be crucial, especially the selection of a reference satellite having a high elevation is essential for good results. The simulation was also used to determine the sweet spot for the antenna baseline length. Additionally the approach allows to experiment with optimising the solution space and stability of the algorithms. Therefore it is a valuable tool for further analysis. Applying the developed approach in the described scenarios for automated driving could create a foundation of higher automation while being independent of the Earth's magnetic field.

IV. Outlook

The constrained dual GNSS approach shown in the previous section is valid when simulated but needs to be verified when used with real measurement data. Using experimental data introduces noise which was neglected in the simulation shown. Consequently no exact solution will be achievable for experimental data which requires optimizing the solution while being aware of potential local minima. Additionally the two independent GNSS receivers need to be synchronized by using the Doppler frequency observations of a corresponding satellite. The synchronisation and general approach of double differencing need to be checked by doing a zero baseline experiment which removes the distance between both antennae by connecting a single antenna to both receivers. The zero baseline experiment should output zero values for all double differences with only small phase residuals and multiples of λ. The simulation results and theory are only valid under the assumption that both receivers sample at exactly the same time, so synchronization is crucial when transitioning towards experimental measurement data.

The scenarios described in the introduction are mostly relevant for automated driving and are especially challenging in urban environments. Therefore combining purely inertial navigation or using a gyroscope for the dual GNSS approach would be beneficial. The combination allows to continuously track heading and also to use each approach to complement the other by checking plausibility and to reject measurement artefacts. For urban environments

it would also help to expand the number of usable satellites by expanding the approach from GPS to other constellations. Having more satellites available creates a basis for even more robust heading estimation since a careful and environment based satellite geometry selection is enabled.

REFERENCES

[1] Blocher, L., Baklanov, F., Hiller, T., Rocznik M., Gerlach, J. and Bringmann O., *An Experimental Localization Sensor Platform for Enhanced Initial Heading Estimation*, 2020 IEEE Inertial Conference.

[2] Blocher, L., Mayer, W., Arena, M., Radović, D., Hiller, T., Gerlach, J. and Bringmann O., *Purely Inertial Navigation with a Low-Cost MEMS Sensor Array*, 2021 IEEE Inertial Conference.

[3] Blocher, L., Mayer, W. ,Vujadinović, M., Haack, J., Hofele, J., Radović, D., Hiller, T., Gerlach, J. and Bringmann O., *Gyroscope-Aided Odometry Navigation Using a Highly-Precise Automotive MEMS IMU Complemented by a Low-Cost Sensor Array*, 2022 IEEE Inertial Conference.

[4] Hiller, T., Vujadinović, M., Blocher, L, Mayer, W., Radović, D, *Practical Approaches to Allan Deviation Analysis of Low-Cost MEMS Inertial Sensors*, 2023 IEEE Inertial Conference.

[5] Hiller, T., Blocher, L., Vujadinovic, M.;, Pentek, Z., Buhmann, A., Roth, Hubert, *Analysis and Compensation of Cross-Axis Sensitivity in Low-Cost MEMS Inertial Sensors*, 2021 IEEE Inertial Conference.

[6] Ben-Ari, M., Mondada, F., *Elements of Robotics*, 2017, Springer Nature.

[7] Bräunl, T. *Embedded Robotics*, 2008, Springer.

[8] van Brummelen, J,, O'Brien, M., Gruyer, D., Najjaran, H., *Autonomous Vehicle Perception: The Technology of Today and Tomorrow*, 2018, Transportation Research Part C: Emerging Technologies.

[9] van Graas, F., Braasch, M., *GPS Interferometric Attitude and Heading Determination: Initial Flight Test Results*, 1991, Navigation.

[10] Bonnifait, P.; Bouron, P.; Crubille, P.; Meizel, D., *Data Fusion of Four ABS Sensors and GPS for an Enhanced Localization of Car-like Vehicles*, 2001, Proceedings 2001 ICRA 2001.

[11] Pentek, Z., Hiller, T., Czmerk, A, *Algorithmic Enhancement of Automotive MEMS Gyroscopes With Consumer-Type Redundancy*, 2021, IEEE Sensors Journal.

[12] Li, W., Fan, P., Cui, X., Zhao, S., Ma, T., Lu, M., *A Low-Cost INS-Integratable GNSS Ultra-Short Baseline Attitude Determination System*, 2018, MDPI Sensors.

[13] Lammertsma, P., *Satellite Navigation*, 2005, Institute of Information and Computing Sciences, Utrecht University.

[14] Van Sickle, J., *GPS for Land Surveyors*, 2015, CRC Press.

Lukas Blocher received his M.Eng. degree in Systems Engineering in 2019. From 2019 to 2020 he was involved in developing localization/AHRS systems for Urban Air Mobility as systems architect. He is currently in the industrial Ph.D. program at Robert Bosch GmbH, Reutlingen, Germany in cooperation with the University of Tuebingen. His focus is on initial heading estimation with inertial and multi-antenna GNSS systems.

Hochschule Düsseldorf
University of Applied Sciences
HSD

Vergleich von UVM-SystemC mit etablierten UVM-Implementierungen

Moritz Kupke, Stephan Gerth, Bernhard M. Rieß

***Zusammenfassung*—Ziel dieser Arbeit war es, die Anwendbarkeit von UVM-SystemC bei der Schaltungsverifikation zu überprüfen. Dazu wurde UVM-SystemC mit den beiden etablierten UVM-Implementierungen UVM-Specman *e* und UVM-SystemVerilog verglichen. Dieser Vergleich wurde anhand von zwei Designs, einem SFIFO und einem Arbiter, durchgeführt. Dazu wurden beide Schaltungen in allen drei UVM-Implementierungen verifiziert. Für die Verifikation wurden EDA-Tools der Firma Cadence Design Systems verwendet. Diese Arbeit kommt zu dem Ergebnis, dass UVM-SystemC im Allgemeinen anwendbar ist, jedoch einzelne Einschränkungen hingenommen werden müssen und stellenweise Verbesserungspotential besteht. Verbesserungspotenzial besteht unter anderem bei dem Tool Support durch die EDA-Tools. Positiv fallen die allgemeinen UVM Funktionen auf, hier ist UVM-SystemC vergleichbar mit den etablierten UVM-Implementierungen.**

***Schlüsselwörter*—Schaltungsverifikation, UVM, UVM-Specman *e*, UVM-SystemVerilog, UVM-SystemC**

I. Einleitung

Die Verifikationsmethodik integrierter Schaltungen hat sich in den letzten Jahrzehnten stark gewandelt. Frühere Methoden sind aufgrund der gestiegenen Komplexität integrierter Schaltungen nicht mehr sinnvoll anwendbar, um eine umfassende Verifikation zu gewährleisten. Inzwischen hat sich die Universal Verification Methodology (UVM) als State-of-the-Art im Bereich Coverage-Driven-Verification durchgesetzt, um die Komplexität zu bewältigen. Diese bietet ein entsprechendes Verifikationsframework, um zusammen mit Constrained Randomization eine entsprechende Verifikationsabdeckung zu erreichen.

Die meistgenutzte Implementierung von UVM ist derzeit UVM-SystemVerilog, gefolgt von UVM-Specman *e*. Als dritte Implementierung wird derzeit UVM-SystemC von der Accellera SystemC Verification Working Group (VWG) [2] entwickelt und steht als öffentliche Proof-of-Concept Implementierung zur Verfügung. UVM-SystemC bietet als Alleinstellungsmerkmal die Unabhängigkeit von kommerziellen Simulatoren und somit eine größere Flexibilität im Austausch von Modellen mit Entwicklungspartnern und Kunden.

Inwieweit UVM-SystemC im Vergleich zu den etablierten UVM-Implementierungen anwendbar ist und welche Kriterien dafür berücksichtigt werden müssen wurde in dieser Arbeit untersucht. Die Kriterien für die Anwendbarkeit wurden aus den üblichen Implementierungen abgeleitet. Dazu zählen beispielsweise die Lizenzierung, die Weitergabe an Entwicklungspartner und Kunden, die Verwendung von Verifikations IPs, die Integration in bisherige Tools oder die Erkennbarkeit von Fehlern. Darüber hinaus wurde analysiert, in welchem Umfang bereits bestehende EDA-Tools für den Einsatz von UVM-SystemC genutzt werden können. Dazu wurden zwei Designs unterschiedlicher Komplexität in den drei untersuchten UVM-Implementierungen verifiziert. Anhand der beiden Designs konnten die Unterschiede zwischen den einzelnen UVM-Implementierungen herausgearbeitet werden. Abschließend wurden die vorab identifizierten Kriterien zur Bewertung der Anwendbarkeit von UVM-SystemC verwendet.

Moritz Kupke, moritz.kupke@bosch.com, Stephan Gerth, stephan.gerth@bosch.com, Bernhard Rieß, bernhard.riess@hs-duesseldorf.de

II. Stand der Technik

UVM ist heutzutage eine etablierte Verifikationsmethodik, welche viele Vorteile bietet, wie z.B. die Wiederverwendbarkeit von Verifikationskomponenten. UVM wird bei vielen Verifikationsprozessen verwendet und profitiert davon, dass es unabhängig von einer Verifikationssprache ist. Dadurch stellt UVM sicher, dass die Entwicklungsergebnisse verschiedener Firmen kompatibel sind, egal welche Verifikationssprache Verwendung findet. Die Methodik wird heutzutage von drei verschiedenen Verifikationssprachen unterstützt: Specman *e*, SystemVerilog und SystemC. Da UVM bei allen Implementierungen Gemeinsamkeiten aufweist, werden diese zunächst im Allgemeinen erläutert und im späteren Teil wird auf die einzelnen Sprachen eingegangen.

A. Aufbau einer UVM-Umgebung

Zur Schaltungsverifikation werden zwei Komponenten benötigt: Das Device under Verification (DUV) und die Verifikationsumgebung (siehe Abbildung 1). Diese werden in der Testbench (TB) verbunden. Die Verbindung zwischen den beiden Komponenten wird über ein sogenanntes Interface (IF) oder eine Signal Map (SMP) realisiert. [1]

Die oberste Ebene der Verifikationsumgebung trägt den Namen Environment (ENV) und bildet das Top

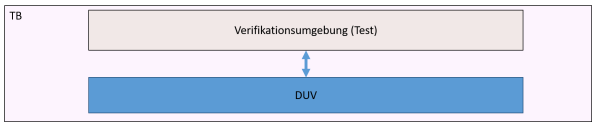


Abbildung 1. Testbench

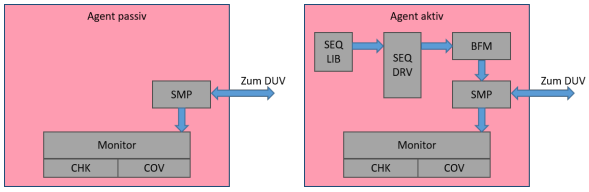


Abbildung 2. passiver Agent li., aktiver Agent re.

Modul der Verifikation. Das ENV instanziiert mindestens einen Agent, welcher ein passiver oder ein aktiver Agent sein kann.
Ein passiver Agent (Abbildung 2 links) übernimmt eine aufzeichnende Funktion und besteht deshalb nur aus einer Komponente, dem Monitor.

Der Monitor greift alle Signale ab, die an einem bestimmten Interface anliegen (blaue Pfeile in Abbildung 2 links) und zeichnet die Kommunikation auf. Zudem kann der Monitor die Coverage aufzeichnen. Coverage bedeutet, dass überprüft wird, ob z.B. ein Signal bestimmte Werte annimmt oder nicht. Man unterscheidet zwischen der Toggle-Coverage (geht ein Signal von 0 auf 1 und umgekehrt), der Finite State Machine (FSM)-Coverage (werden alle Zustände einer Schaltung erreicht), der Code-Coverage (wird jede Zeile mindestens einmal ausgeführt) und der Assertion-Coverage (werden alle Assertions tatsächlich überprüft). Zudem ist es möglich, Checks für die Überprüfung bestimmter Eigenschaften, die das DUV erfüllen muss, aufzurufen.

Ein aktiver Agent (Abbildung 2 rechts) hingegen ergänzt den passiven Agent um einen Sequence Driver (SEQ-DRV), eine Sequence Driver Library (SEQ-LIB) und ein Bus Functional Model (BFM).

Aufgabe des SEQ-DRV ist es Test-Sequenzen (Test-Pattern) zu erzeugen, die dann über das BFM an die Signal Map gelangen (blaue Pfeile in Abbildung 2 rechts) und das DUV stimulieren. Die Test-Sequenzen werden in einem zusätzlichen Modul, der sogenannten Sequence Driver Library (SEQ-LIB), erstellt, gespeichert und vom SEQ-DRV aufgerufen. Zusätzlich zur SEQ-LIB ist es möglich ein Sequenz Item zu verwenden, welches alle relevanten Variablen instanziiert.

Neben dem Agent kann die ENV ein Scoreboard (SCRBRD) zur Überprüfung einbinden wie in Abbildung 3 dargestellt. Dessen Aufgabe besteht darin, die Daten aus den Monitoren zu vergleichen. Empfängt das Scoreboard von der Spezifikation abweichende Daten, wird eine Fehlermeldung angezeigt. Die Daten des Monitors werden über den sogenannten Transaction

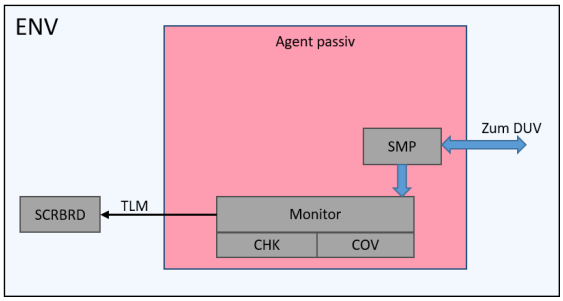


Abbildung 3. UVM ENV mit Agent und Scoreboard

Level Modeling Port (TLM-Port) übertragen (schwarze Pfeile in Abbildung 3). Der TLM-Port eignet sich für die Kommunikation zwischen einzelnen Modulen. Die gleichzeitige Verbindung zwischen mehreren Modulen ist möglich.

Zusätzlich ermittelt das Scoreboard eine Coverage, um weitere Fehler zu erkennen bzw. die Testabdeckung zu überprüfen. Zusätzlich zu Scoreboard und Agent kann die ENV weitere ENV instanziieren wie in Abbildung 4 dargestellt.

Für die automatisierte Erstellung von Komponenten wie z.B. des Agents kann eine Konfiguration (Config) verwendet werden. Diese erstellt dem Entwickler wiederkehrende Codestrukturen und spart somit Zeit. Gibt es in der Testbench mehrere Environments oder Agents, so muss zusätzlich gewährleistet werden, dass alle SEQ-DRV synchron arbeiten [1]. Diese Synchronisierung übernimmt der Virtual Sequence Driver (VSEQ). Der VSEQ ist mit allen untergeordneten Sequence Drivern verbunden (rote Pfeile in Abbildung 4) und kann in jedem verbundenem SEQ-DRV einzelne Sequenzen starten. Um mehrere verschiedene Szenarien (Eingangsfolgen am DUV) zu erstellen, kann sich der VSEQ aus der Virtual Sequence Driver Library (VSEQ-LIB) bedienen. Dort sind alle möglichen Verknüpfungen der jeweiligen Sequence Driver hinterlegt.

Damit verschiedene Test-Szenarien erstellt werden können, gibt es die letzte Hierarchieebene, den Testcase (Test in Abbildung 4). Die Aufgabe des Testcases ist es, verschiedene VSEQ zu verknüpfen und zu starten. Zusätzlich legt der Testcase Start und Ende des Testes fest. Zur Verdeutlichung, dass eine ENV weitere Environments instanziieren kann, wurden in Abbildung 4 exemplarisch zwei Environments eingebunden.

B. *UVM-Specman e*

UVM-Specman *e* ist eine Hardwareverifikationssprache (HVL), welche 1992 von Yoav Hollander ins Leben gerufen wurde. Weiterentwickelt wurde sie von der Firma Verisity, welche später von der Firma Cadence Design Systems (Cadence) aufgekauft wurde. UVM-Specman *e* zeichnet sich dadurch aus, dass sie speziell für die Verifikation entwickelt wurde. Das heißt mit der Sprache kann kein Design modelliert

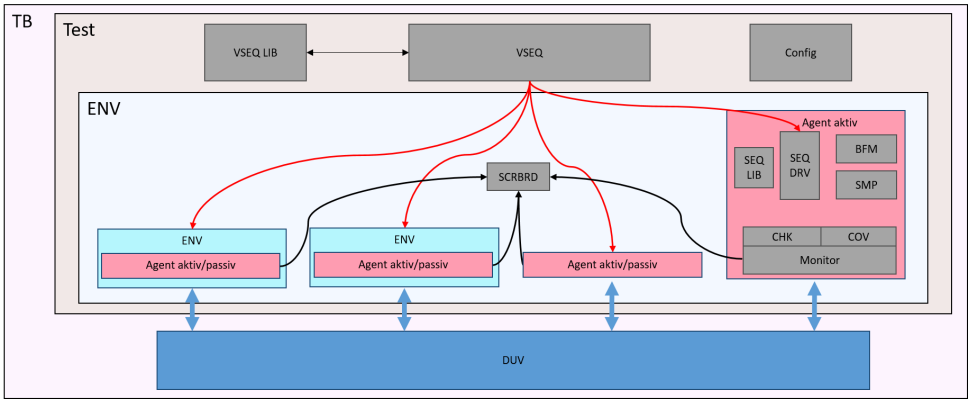


Abbildung 4. UVM Umgebung

werden, jedoch kann die Verifikation von Designs in allen gängigen Hardwarebeschreibungssprachen, wie z.B. Verilog, VHDL oder SystemC, erfolgen. *e* ist eine aspektorientierte Programmiersprache (AOP). Damit bei dieser Sprache die Effizienz und Wiederverwendbarkeit gegeben ist, wurde für sie die *e*RM (*e* Reuse Methodology) eingeführt. Diese soll zudem die Verifikationszeiten, durch u.a. den Re-use von Komponenten, verkürzen. Mit der Gründung von Accellera wurde dann die UVM-Methodik für *e* adaptiert und bis heute verwendet.

C. UVM-SystemVerilog

Im Gegensatz zu UVM-Specman *e* ist UVM-SystemVerilog keine reine HVL, denn bei UVM-SystemVerilog handelt es sich um eine Weiterentwicklung der Hardwarebeschreibungssprache (HDL) Verilog (IEEE 1364). UVM-SystemVerilog wurde von Accellera entwickelt und in der IEEE 1800 standardisiert. Dank der Entwicklung durch Accellera ist UVM-SystemVerilog eine frei verfügbare Sprache und es gibt frei verfügbare Dokumentationen und Beispiele [1]. Im Gegensatz z.B. zu UVM-Specman *e* ist UVM-SystemVerilog nicht an kommerzielle Simulatoren gebunden und darüber hinaus eine objektorientierte Programmiersprache (OOP). Wie auch UVM-Specman *e* unterstützt UVM-SystemVerilog die UVM Methodik. Bei UVM-SystemVerilog ist UVM als Klassenbibliothek eingebunden und kann so Verwendung finden.

D. UVM-SystemC

UVM-SystemC ist die neueste UVM-Implementierung von Accellera und in der IEEE 1666 standardisiert [2]. UVM-SystemC ist keine reine HVL, sondern ergänzt die SystemC HDL Implementierung. Wie für UVM-SystemVerilog wurde für UVM-SystemC eine UVM-Klassen Bibliothek angelegt. UVM-SystemC ist zudem keine eigene „Programmiersprache“, sondern eine Erweiterung der objektorientierten Sprache C++. Dies bietet den Vorteil, dass keine kommerziellen Simulatoren verwendet werden müssen sondern ein frei verfügbarer C++-Compiler genügt. Weiterhin kann die Beschreibung in einer höheren Abstraktionsebene erfolgen, wie z.B. durch Verwendung von TLM Ports. Durch viele verfügbaren Dokumentationen und Beispiele lässt sich die Verifikationsumgebung zudem leicht adaptieren. Hinzu kommt, dass UVM-SystemC DUVs in allen gängigen HDLs unterstützt.

III. Vergleich und Anwendung der UVM-Implementierungen

Für den Vergleich der drei verschiedenen UVM-Implementierungen wurden zunächst zwei repräsentative Designs indentifiziert, die im Folgenden kurz charakterisiert werden. Anschließend wurden die anzuwendenden Vergleichskriterien definiert und mit Hilfe einer Entscheidungsmatrix untereinander priorisiert. Dann wurde ein Verifikationsplan entwickelt um einen einheitlichen Vergleich zwischen den drei UVM-Implementierungen zu gewährleisten und damit die einzelnen Vor- und Nachteile herausarbeiten zu können. Dazu wird auch die verwendete Entwicklungsumgebung kurz beschrieben. Abschließend werden die Ergebnisse des Vergleichs diskutiert.

A. Testcases

Im Rahmen dieser Arbeit wurden zwei unterschiedliche Designs mit den drei UVM-Implementierungen verifiziert. Die beiden Designs lagen jeweils in der Beschreibungssprache VHDL vor und wurden in allen drei UVM-Implementierungen in VHDL verifiziert.

Als erstes Modul wurde ein **SFIFO** (Synchronous First In First Out Buffer) gewählt (vgl. Abbildung 5. Das SFIFO ist ein einfaches DUV und sollte einen schnellen und einfachen Überblick über die Anwendbarkeit von UVM-SystemC geben. Es verfügt über 9 Ein- und Ausgänge. Diese lassen sich in drei funktionale Einheiten unterteilen. Die erste Einheit (FIFO Main) stellt das Taktsignal (clk) und den asynchronen Reset (reset_n). Die zweite Einheit (WRITE) ist für das Schreiben von Daten in das FIFO zuständig und setzt

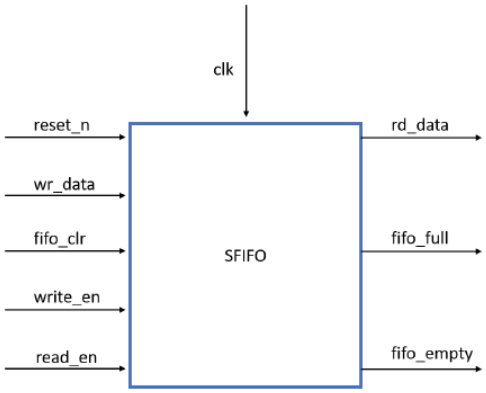


Abbildung 5. SFIFO

sich aus den Logik-Signalen write enable (write_en), write Data (wr_data) und fifo full (fifo_full) zusammen. Ist das write_en Signal auf high gesetzt und liegt am Taktsignal eine steigende Taktflanke an, so werden die Daten über die wr_data Bus Leitung in den FIFO geschrieben. Die Bus Leitung kann eine variable Breite von 1 bis 256 Bit haben. Über das Signal fifo_full wird dem Anwender signalisiert, ob der FIFO voll ist (das Signal ist high) oder nicht (das Signal ist low). Die Änderung des Signals wird immer mit der nächsten steigenden Taktflanke ausgegeben. Zusätzlich ist die Anzahl der Speicherplätze fest einstellbar. Als dritte Einheit (READ) gibt es die Lese-Einheit, die für das Auslesen des FIFOs zuständig ist. Dafür signalisiert ein read enable Signal (read_en) bei einem positiven Wert und einer steigenden Taktflanke dem FIFO, dass Daten ausgelesen werden sollen. Die Daten liegen mit der nächsten steigenden Taktflanke an der read Datenleitung (rd_data) an. Die Datenleitung hat die selbe Breite wie die wr_data Leitung. Damit der Anwender informiert wird, wenn der FIFO leer ist, also keine Daten mehr im Speicher sind, gibt es das fifo_empty Signal welches auf high geht, wenn der FIFO leer ist.

Als zweites Modul wurde ein **Arbiter** verwendet. Der Arbiter ist ein komplexeres und aufwendigeres Modul als der SFIFO. Er verfügt über 3 APB-Masterinterfaces und 4 APB-Slaveinterfaces, sowie einer Logik zum Verteilen der Zugriffe der Masters auf die Slaves. Der Arbiter soll die Kommunikation zwischen drei Master-Modulen und vier Slave-Modulen steuern. Die Kommunikation ist bidirektional d.h., es ist möglich Daten vom Master zum Slave zu senden und Daten vom Slave zum Master zu übertragen. Alle Module kommunizieren nach dem AMBA 3 APB Protokoll von ARM Limited. Die Module sind nach folgendem Schema verbunden, wie in Abbildung 6 zu sehen:

Master 0 kann mit Slave 0 und 1 kommunizieren (blaue Pfeile). Master 1 ist mit Slave 0, 1 und 2 verbunden (grüne Pfeile). Master 3 kommuniziert mit Slave 1, 2 und 3 (rosa Pfeile). Hierbei muss beachtet

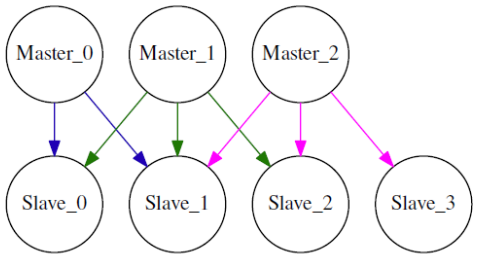


Abbildung 6. Arbiter.

werden, dass die Master-Module gleichzeitig kommunizieren können. Jedoch darf es nicht passieren, dass beispielsweise Master 2 und Master 1 gleichzeitig auf Slave 2 zugreifen. Dafür ist eine Priorisierung in der Schaltung vorgesehen, dies bedeutet, dass der Master mit der höheren Priorität Vorrang erhält. In der konkreten Anwendung soll Master 0 die höchste Priorität bekommen und Master 2 die geringste Priorität. In dem vorherigen Beispiel würde dies bedeuten, das Master 1 den Vorrang erhält und Master 2 warten muss.

Der Arbiter bot den Vorteil, dass komplexere Probleme, wie z.B. die Verwendung der Konfigurierbarkeit oder des UVM-Debuggings, intensiv analysiert werden konnten.

B. Vergleichskriterien

Für den Vergleich der drei UVM-Implementierungen wurde eine Vielzahl von Kriterien herausgearbeitet, die im Folgenden beschrieben werden. Sie finden dann im weiteren Vergleich auf beide Testcases Anwendung.

- **DUV-Language**
  Beschreibt in welchen Sprachen DUVs modelliert sein können, um mit der jeweiligen UVM-Implementierung verifiziert werden zu können.
- **TLM-Kommunikation**
  Beschreibt die Kommunikation zwischen einzelnen Modulen. Wie z.B. zwischen dem Monitor und dem Scoreboard.
- **Konfigurierbarkeit**
  Ist ein Modul über bestimmte Variablen anpassbar?
- **Constrained Randomization**
  Wie funktioniert die Randomisierung bzw. die Einschränkung der Randomisierung?
- **Coverage**
  Wie kann die Coverage erzeugt und gespeichert werden? Gibt es Unterschiede beim Erstellen der Coverage?
- **Checks**
  Können Checks verwendet werden?
- **Lines of Code (LOC)**
  Wie viele Zeilen Programmcode im Simulationsskript sind nötig?

- **UVM-Methoden**
  Werden alle UVM-Methoden bzw. -Funktionen unterstützt?
- **Templates**
  Können Templates zur Erstellung der Verifikationsumgebung verwendet werden?
- **Wiederverwendung (Re-use)**
  Welche Komponenten können wiederverwendet werden? An welchen Stellen im Verifikationscode müssen Änderungen vorgenommen werden?
- **Verification IPs (VIPs)**
  Gibt es wiederverwendbare Verifikations IPs? Welche Hersteller stellen VIPs zur Verfügung?
- **Weitergabe an den Kunden**
  Ist es möglich die fertige Testbench in einer Form an Entwicklungspartner und Kunden weiterzugeben, so dass sie dort gut genutzt werden kann?
- **EDA-Tool-Support**
  Gibt es Unterstützung von EDA-Tools? Hier wurde die Unterstützung für die UVM-Implementierungen mit Tools der Fa. Cadence geprüft.
- **Code Debugging im EDA-Tool**
  Wie lassen sich Fehler im DUV finden?
- **UVM-Debugging**
  Wie können einzelne UVM-Methoden im Simulator angezeigt werden?
- **Anzeigen bzw. Ausblenden von Konsolenausgaben**
  Können Info- oder Fehlermeldungen angezeigt oder ausgeblendet werden?
- **Lizenzkosten**
  Für welche der eingesetzten Tools fallen Lizenzkosten an?
- **Simulationszeit**
  Beschreibt die Zeit, die für die Ausführung der Simulation am Rechner benötigt wird.
- **UVM-Multi-Language (UVM-ML)**
  Ist es möglich DUVs, die in verschiedenen Hardwarebeschreibungssprachen modelliert sind, zu verifizieren?
- **Regression**
  Beschreibt das Handling von mehreren Testcases, sowie die Auswertung.
- **Ein- und Ausblenden von Coverpunkten**
  Kann die Coverage-Anzeige im Nachhinein verändert werden?

In Summe wurden somit 21 Kriterien erarbeitet, anhand derer die Anwendbarkeit von UVM-SystemC im Vergleich zu UVM-Specman *e* und UVM-SystemVerilog im Folgenden praktisch analysiert werden soll.

C. Erstellung einer Entscheidungsmatrix

Um eine Prioritätsreihenfolge der einzelnen Vergleichskriterien festzulegen zu können wurde eine Entscheidungsmatrix entwickelt. In dieser Entscheidungsmatrix wurde die Priorität jedes einzelnen Kriteriums mit jedem anderen Kriterium verglichen und daraus eine Gesamtpunktzahl für jedes einzelne Kriterium ermittelt, welche dessen Bedeutung und damit Priorität definiert. Dazu wurden alle Kriterien in die Zeilen und Spalten der Entscheidungsmatrix eingetragen wie beispielhaft in Tabelle I dargestellt.

Tabelle I
BEISPIEL ENTSCHEIDUNGSMATRIX.

Kriterium	A	B	C	Gesamt
A	x	2	1	3
B	0	x	0	0
C	1	2	x	3

In diesem Beispiel soll Kriterium A wichtiger sein als Kriterium B, deshalb wird in die entsprechende Zelle (Zeile A, Spalte B) eine 2 eingetragen. Weiterhin sind A und C gleich wichtig, daher wird in die Zelle (A, C) eine 1 eingetragen. Ist B weniger wichtig als C, wird in Zelle (B, C) eine 0 eingetragen. Die Zellen unterhalb der Diagonale der Matrix ergeben sich durch das „Umkehren“ der Werte oberhalb der Diagonale. So wird beispielsweise der Wert für die Zelle (B, A) berechnet durch $W_{B,A} = 2 - W_{A,B} = 2 - 2 = 0$.

Die Entscheidungen in Tabelle II wurden für alle 21 Kriterien nach der allgemeinen Auffassung durchgeführt und sollten für die meisten Anwendungsfälle gültig sein. Falls der Anwender eigene Gewichtungen bzw. Entscheidungen vornehmen möchte, kann dies natürlich in einer modifizierten Tabelle berücksichtigt werden. Sind alle Entscheidungen durchgeführt, ergibt sich am Ende durch die Addition der Werte in allen Spalten einer Zeile die Gesamtpunktzahl für jedes einzelne Kriterium. Die Gesamtpunktzahl wurde dann als Indikator für die Priorität der einzelnen Kriterien verwendet. Aus der Matrix kann abgeleitet werden, dass die Lizenzkosten mit 40 Punkten die höchste Punktzahl erreichen und somit das wichtigste Kriterium darstellen. Die niedrigste Punktzahl (3) erreicht die Weitergabe an den Kunden.

Basierend auf den mit Hilfe dieser Entscheidungsmatrix ermittelten Gesamtpunktzahlen für jedes Kriterium werden dann in Kapitel IV der Ergebnisse des Vergleichs der drei UVM-Implementierungen bewertet und verglichen.

D. Verifikationsplanung und -durchführung

Um die einzelnen Unterschiede und Gemeinsamkeiten der drei UVM-Implementierungen praxisnah vergleichen zu können wurde ein Verifikationsplan erstellt und abgearbeitet. Für eine erfolgreiche Verifikation eines DUV hat sich der folgende sechsschrittige Prozess etabliert:

1. Erstellen eines Verifikationsplans (VPlan):
Der VPlan gliedert sich in drei Einheiten. Die erste

Tabelle II
ENTSCHEIDUNGSMATRIX

Kriterien	DUV Language	TLM-Kommunikation	Konfigurierbarkeit	Randomisierung	Coverage	Checks	LOC	UVM-Methoden	Templates	Wiederverwendung	Verification IPs	Kunden	EDA-Tool-Support	Code im EDA-Tool	UVM-Debugging	Konsolenausgaben	Lizenzkosten	Simulationszeit	UVM-Multi-Language	Regression	Coverage-Anzeige	**Gesamtpunktzahl**
DUVLanguage	x	1	2	2	2	2	1	0	1	0	1	1	1	0	0	1	0	0	1	1	1	**18**
TLM-Kommunikation	1	x	2	1	1	1	2	1	2	1	2	1	1	1	1	2	0	1	1	1	2	**25**
Konfigurierbarkeit	0	0	x	0	0	0	1	1	2	1	1	2	1	2	2	1	0	2	1	2	0	**19**
Randomisierung	0	1	2	x	1	1	2	2	2	2	2	2	2	2	2	2	0	2	2	1	2	**32**
Coverage	0	1	2	1	x	2	2	2	2	2	2	2	2	2	2	2	0	2	2	1	1	**32**
Checks	0	1	2	1	0	x	2	2	2	2	2	2	2	2	2	2	0	2	2	1	1	**30**
LOC	1	0	1	0	0	0	x	1	1	0	1	2	0	0	0	1	0	1	1	0	0	**10**
UVM-Methoden	2	1	1	0	0	0	1	x	1	1	1	2	1	1	1	1	0	0	2	0	0	**16**
Templates	1	0	0	0	0	0	1	1	x	0	1	2	0	0	0	0	0	0	2	0	0	**8**
Wiederverwendung (Re-use)	2	1	1	0	0	0	2	1	2	x	2	2	1	1	1	2	0	1	2	1	2	**24**
Verification IPs	1	0	1	0	0	0	1	1	1	0	x	1	0	0	0	0	0	0	1	0	0	**7**
Weitergabe an Kunden	1	1	0	0	0	0	0	0	0	0	1	x	0	0	0	0	0	0	0	0	0	**3**
EDA-Tool-Support	1	1	1	0	0	0	2	1	2	1	2	2	x	1	2	1	0	0	2	1	1	**21**
Code im EDA-Tool	2	1	0	0	0	0	2	1	2	1	2	2	1	x	1	2	0	2	2	1	2	**24**
UVM-Debugging	2	1	0	0	0	0	2	1	2	1	2	2	0	1	x	2	0	1	2	1	1	**21**
Konsolenausgaben	1	0	1	0	0	0	1	1	2	0	2	2	1	0	0	x	0	0	2	0	0	**13**
Lizenzkosten	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	x	2	2	2	2	**40**
Simulationszeit	2	1	0	0	0	0	1	2	2	1	2	2	2	0	1	2	0	x	2	1	2	**23**
UVM-Multi-Language	1	1	1	0	0	0	1	0	0	0	1	2	0	0	0	0	0	0	x	0	0	**7**
Regression	1	1	0	1	1	1	2	2	2	1	2	2	1	1	1	2	0	1	2	x	2	**26**
Coverage-Anzeige	1	0	2	0	1	1	2	2	2	0	2	2	1	0	1	2	0	0	2	0	x	**21**

Einheit legt fest, welche Ein- und Ausgangssignale eines DUV gecovert werden sollen (Coverage). die zweite Einheit definiert, welche Szenarien überprüft werden sollen (Checks). Dazu wird die DUV-Spezifikation herangezogen. Die letzte Einheit ist die Code-Coverage. Zur Implementierung von Verifikationsplänen gibt es Tools der EDA-Hersteller, um diese am Ende des Prozesses analysieren und in der Regression wiederverwenden zu können.

2. Erstellen der Verifikationsumgebung:
Ist der VPlan abgeschlossen, folgt die konzeptionelle Erstellung der Verifikationsumgebung. Diese wird zunächst graphisch entwickelt und sollte alle benötigten Komponenten, wie in Kapitel II beschrieben enthalten. Hierbei ist es empfehlenswert, die einzelnen Environments und Agents in Funktionsgruppen zu gliedern.

3. Implementierung der Verifikationskomponenten:
Nach Fertigstellung der konzeptionellen Verifikationsumgebung kann mit der Implementierung begonnen werden. Die größten Unterschiede zwischen den drei UVM-Methodiken treten bei ihrer Implementierung auf. Es wird empfohlen für jede UVM-Komponente eine eigene Datei anzulegen und mittels geeigneter Ordner zu strukturieren.
Mit Hilfe des DUV SFIFO wurde analysiert, welche Templates zur Verfügung stehen. Insbesondere sollten hier die SEQ-LIB mit der Randomisierung, das Scoreboard mit der TLM-Kommunikation zu den Monitoren, die Coverage, die Checks und die Konfigurierbarkeit überprüft werden. In diesem Zusammenhang ist es sinnvoll zu erörtern, welche Komponenten wiederverwendet werden können.
Um einen weiteren Überblick über die Anzahl der code-relevanten Zeilen (Lines of Code (LOC)) zu bekommen, zählt das Tool `Cloc` [8] die Anzahl der wirksamen Codezeilen aber nicht die Kommentare oder die Länge der Zeilen.
Die Testbench ruft die Testcases auf. Dazu floß der Aufruf von Sequenzen in den Vergleich ein. Zudem wurde verglichen, wie der Aufruf der Testcases im Detail funktioniert.

4. Erstellung von Testszenarien:
Ist die Implementierung der Verifikationskomponenten vollendet, folgt die Erstellung von Testszenarien. Dazu werden oft mehrere verschiedene Randbedingungen sog. "Cornercases" und Standardbedingungen getestet. Zusätzlich können zufällige Eingangsfolgen getestet

werden, um schwer zu findende Fehler aufzudecken.

5. Simulation:
Nach dem Fertigstellen der Testszenarien, wurden die Eigenschaften der Simulatoren untersucht. Mit dem DUV SFIFO wurden insbesondere die Anzeigeeigenschaften des Simulators überprüft. Mit Hilfe des DUV Arbiter wurde überprüft, wie lange die UVM-Implementierungen brauchen, um den ersten Fehler zu finden. Dazu wurden im Arbiter bewusst mehrere Fehler eingebaut. Weiterhin wurden in beiden DUVs die UVM-Debugging Möglichkeiten im Simulator verglichen. Bei beiden DUVs wurde analysiert, ob es mit allen drei UVM-Implementierungen möglich ist, die o.g. Fehler zu finden.
Um dem Anwender einen Überblick über die Fehlerinformationen des Simulators geben zu können, wurden in die SFIFO UVM Umgebung gezielt Fehler eingebaut. Damit wurde geprüft, wie genau die Fehlerinformationen sind. Zusätzlich wurde die benötigte Simulationszeit für beide DUVs untersucht und mit der Kommandooption `-status` im Xcelium Simulator gemessen. Damit das Erstellen im Dateisystem nicht mit in die Simulationszeit mit einfließt, wurde nur die Zeit im Simulator `xmsim` gemessen.

6. Durchführung einer Regression:
Abschließend mussten alle Punkte des VPlans erfüllt sein. Hier wurden auch die Ergebnisse aus dem Simulator mit einbezogen. Dazu wurde in einem Regression-Tool (Auswerteeinheit) der Verifikationsplan mit den Testergebnissen verknüpft und überprüft, ob alle Punkte erreicht wurden. Wichtig bei der Regression sind die Anzeigemöglichkeiten von Tests, sowie Coverage und Assertions (Checks). Schlagen z.B. Testcases fehl oder ist noch nicht überall eine Coverage von 100% erreicht, wird das Verfahren zur Verifikationsplanung und UVM-Implementierung noch einmal von Anfang an durchlaufen. Hinzu kommt die Reparatur von identifizierten Fehlern im DUV. Damit alle Fehler schnell gefunden werden, wird das “Vier Augen Prinzip“ empfohlen. Das bedeutet, dass der Verifikationsprozess und das Verifikationsergebnis von einer zweiten Person überprüft wird. Da die meisten Regression-Tools die selben Simulatoren verwenden, spielt die Zeit hier eine geringe Rolle, sollte aber trotzdem Beachtung finden. Zudem wurde die Möglichkeit des Multiplexings (paralleles Simulieren mehrerer Testcases) bewertet.
Nach Abschluss aller Phasen sollte das Regression-Tool in allen Kategorien 100% anzeigen. Ist dies erreicht ist die Verifikation abgeschlossen und das DUV kann freigegeben werden.

Tabelle III
TOOL-UMGEBUNG.

Tool	Version	Hersteller	Beschreibung
RHEL	6	Red Hat	Betriebssystem
vPlanner	18.09-s004	Cadence	Verifikationsplan
vManager	18.09-s004	Cadence	Regression-Tool
DVT-Eclipse	19.1.40	AMIQ EDA	Editor
Xcelium	18.09.007	Cadence	Simulator
GCC Compiler	6.3.0	GNU-Projekt	Compiler für u.a. C++
SystemC Biblio.	2.3.3	Accellera	UVM-SystemC
UVM-Bibliothek	1.0-beta2	Accellera	UVM-SystemC
SCV Bibliothek	2.0.1	Accellera	SystemC Verifikation
FC4SC	2.1.1	AMIQ-Const.	Coverage Erzeugung

E. Beschreibung der Entwicklungsumgebung

Vor dem Vergleich der drei UVM-Implementierungen wird in diesem Kapitel die genutzte Softwareumgebung beschrieben. Als Betriebssystem wurde RHEL 6 verwendet. Der Verifikationsplan für alle drei UVM-Implementierungen wurde mit dem vPlanner der Firma Cadence erstellt. Für das Starten und Auswerten von Regressions wurde das Tool vManager der Fa. Cadence verwendet. Zum Editieren des Quellcodes fand das DVT-Eclipse 19.1.40 [9] Verwendung.
Für UVM-Specman *e* und UVM-SystemVerilog wurde als Simulator Xcelium 18.09.007 [10] der Firma Cadence (im Folgenden Xcelium) verwendet. Der Simulator stellt die entsprechenden UVM-Bibliotheken für die beiden UVM-Implementierungen bereit. Für UVM-SystemC wurde ein von der Firma Cadence abgewandelter GCC Compiler, Version 6.3.0, verwendet. Hierzu wurde die UVM-SystemC Bibliothek [11], sowie die UVM-Bibliothek [2], die SCV-Bibliothek [4] und die `FC4SC`-Bibliothek [6] eingebunden. Aufgrund der fehlenden Möglichkeit des gemeinsamen Simulierens von VHDL DUVs und UVM-SystemC im GCC Compiler, wurde der Xcelium Simulator verwendet. Der Simulator ermöglicht gemeinsames Simulieren von VHDL und UVM-SystemC.
Jedoch gibt es beim Xcelium Simulator einen entscheidenden Nachteil: Das abgewandelte UVM-SystemC implementiert keine Sequenzen [12, vgl. S. 10]. Deshalb muss die UVM-Bibliothek vom User eingebunden werden und die Coverage Bibliothek `FC4SC` muss wie die UVM-Bibliothek vom User initialisiert werden. Der User muss dadurch auf die UVM-Debugging Funktionen verzichten und er verliert damit die Möglichkeit die Coverage in Tools der Firma Cadence anzeigen zu können. Die Entscheidung für UVM-SystemVerilog und UVM-SystemC fiel auf die Tools der Fa. Cadence, da diese auch für UVM-Specman *e* verwendet werden. Die Hersteller Siemens EDA (ehemals Mentor Graphics) [13] und Synopsys Inc. (Synopsys) [14] bieten zwar Support für UVM-SystemVerilog und UVM-SystemC, dieser wurde jedoch in dieser Arbeit nicht überprüft. In Tabelle III sind zur Übersicht alle Tools und ihre genutzten Versionen aufgelistet.

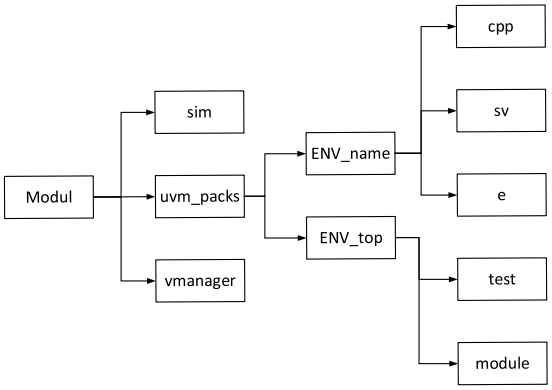


Abbildung 7. Ordnerstruktur

Dateimanagement:
Für ein erfolgreiches und systematisches Datenmanagement, wurden die Dateien nach folgendem Schema benannt: UVM-Specman *e* Dateien enden mit einem `.e`. UVM-SystemVerilog Dateien mit einem `.sv` und bei UVM-SystemC wurde `.cpp` als Endung für die Hauptdatei und `.h` für alle anderen Dateien verwendet.

Die Ordnerstruktur für jedes `Modul` wurde wie in Abbildung 7 dargestellt erstellt: In einem Ordner mit dem Namen `sim` wurden alle Skripte und Daten für die Simulation abgelegt. Daneben wurde ein Ordner `uvm_packs` zur Aufbewahrung aller Quellcode-Dateien angelegt. In diesem sind folgende Unterordner enthalten: Der Unterordner `ENV_name` enthält drei weitere Unterordner, die auf die UVM-Implementierung hinweisen: `e` für UVM-specman *e*, `sv` für UVM-SystemVerilog, `cpp` für UVM-SystemC. In diesen Unterordnern befinden sich dann die jeweiligen Quellcode Dateien. Im Unterordner `ENV_top` ist neben der UVM-Implementierung noch ein Unterordner mit den Testcases (`test`) und ein Unterordner `module` für das zu verifizierende DUV enthalten. In Abbildung 7 wurden aus Gründen der Übersichtlichkeit bei `ENV_top` die Ordner für die UVM-Implementierung weggelassen. Gibt es neben den beiden Environments weitere, so wurde für diese Environments analog zu dem Ordner `ENV_name` vorgegangen. Der Ordner `vmanager` wurde für die Regression Umgebung verwendet.

Ausführen der Simulation und Regression:
Die Ausführung der Simulation und Regression erfolgte für die jeweilige UVM-Implementierung skriptbasiert. Die Grundgerüste dieser Skripte sind für beide DUVs gleich und unterschieden sich nur durch die Übergabeparameter des Simulatoraufrufs.

F. Vergleich der drei UVM-Implementierungen

Ist das Environment für die Verifikation aufgesetzt kann mit der Evaluierung entsprechend dem oben entwickelten Evaluierungsplan begonnen werden. Zunächst wurde der Plan auf das DUV SFIFO und im Anschluss daran auf das DUV Arbiter angewendet. Hierfür wurden die Schritte entsprechend dem Evaluierungsplan nacheinander ausgeführt. Um die drei UVM-Implementierungen miteinander vergleichen zu können, wurden diese anhand der in Kapitel III-B entwickelten 21 Kriterien zahlenmäßig bewertet. Hierfür wurden ganzzahlige Werte von 0 bis 10 vergeben wobei 0 die schlechteste und 10 die beste Bewertung ist. Im Folgenden werden zunächst die drei einzelnen UVM-Implementierungen bezüglich der 21 Kriterien diskutiert und zahlenmäßig bewertet:

DUV-Language:
Es besteht die Anforderung, dass alle drei UVM-Implementierungen Designs in den HDLs VHDL, Verilog und SystemC verifizieren können. Hierbei wurde auch die Abhängigkeit vom jeweiligen Simulator überprüft. Für Specman *e* wie auch für UVM-SystemVerilog wurden 10 Punkte vergeben, da beide UVM-Implementierungen die Möglichkeit bieten alle drei HDLs zu verifizieren. Bei der Verifikation mit UVM-SystemC stellte sich heraus, dass UVM-SystemC alle Hardwarebeschreibungssprachen unterstützt, jedoch gibt es eingeschränkte Funktionen bei DUVs in VHDL und Verilog. Dies kann durch Installation der vollständigen Bibliothek vom Nutzer selbst kompensiert werden. Dabei gehen allerdings die UVM-Debugging Funktionen verloren, wie sie in UVM-SystemVerilog und UVM-Specman *e* unterstützt wurden. Die anderen UVM-Implementierungen zeigen hier z.B. die Namen der einzelnen Signale an. Für UVM-SystemC wurden daher nur 5 Punkte vergeben.

TLM-Kommunikation:
Bezüglich der TLM-Kommunikation unterstützen alle drei UVM-Implementierungen die selben Kommunikationswege, jedoch unterscheiden sie sich im Aufruf der Funktionen. So kann bei UVM-Specman *e* ein Funktionsname festgelegt werden. Bei UVM-SystemVerilog kann der Funktionsname auch erweitert werden. Daher wurden für UVM-Specman *e* und UVM-SystemVerilog jeweils 10 Punkte vergeben. Bei UVM-SystemC ist die Funktion nur abhängig von dem übergebenden Parameter bzw. es kann durch weiteres TLM-Handling der Funktionsname verändert werden. Dafür muss eine neue Klasse instanziiert werden, der UVM-Subscriber. Das Einbinden des Subscribers benötigt zusätzliche Codezeilen im Skript, die bei UVM Specman *e* und UVM-SystemVerilog nicht benötigt werden. Jedoch kann durch Einsatz eines Templates dieser Aufwand reduziert werden. Zudem gibt es für UVM-SystemC nicht direkt die Möglichkeit im Scoreboard verschiedene TLM-Funktionen aufzurufen. Daher wurden für UVM-SystemC 8 Punkte vergeben.

Konfigurierbarkeit:
Beim Kriterium Konfigurierbarkeit wurden beim DUV SFIFO und beim DUV Arbiter keine Unterschiede zwischen den drei UVM-Implementierungen festgestellt. Beim DUV SFIFO gab es in allen UVM-Implementierungen die Möglichkeit die Agents passiv oder aktiv zu verwenden. Im DUV Arbiter war es zudem möglich den Namen des jeweiligen Master oder Slaves mittels der Konfiguration festzulegen. Aus diesem Grund erhielten alle drei UVM-Implementierungen die beste Bewertung (10 Punkte).

Constrained Randomization:
Bei der Randomisierung lassen sich in allen drei UVM-Implementierungen Constraints nutzen, jedoch können diese Constraints bei UVM-SystemVerilog nicht im Testcase festgelegt werden. Dafür müssen der SEQ-LIB Parameter übergeben werden. Dies erfordert mehr Aufwand (9 Punkte). Bei UVM-Specman *e* und UVM-SystemC ist die Übergabe aus dem Testcase möglich, jedoch ist die benötigte Zeit bis zur Fehleridentifikation deutlich größer (jeweils 9 Punkte). Voraussetzung dafür ist bei beiden Methoden, dass die Variablen in der SEQ-LIB deklariert werden. Bei der Einschränkung der Werte gibt es keine Unterschiede, dort können die 3 UVM-Implementierungen alle Einschränkungen vornehmen.

Coverage:
Für die Erzeugung von Coverage-Punkten stellen die UVM-Implementierungen UVM-Specman *e* und UVM-SystemVerilog Methoden zum Aufsammeln von Coverage zur Verfügung. Bei UVM-SystemC muss zusätzlich die `FC4SC` Bibliothek eingebunden werden. Dadurch werden Coverage Funktionen wie bei UVM-SystemVerilog erhalten und es lassen sich alle Coverage Punkte aufsammeln. Dazu können Funktionen wie z.B. `illegal bin` und `exclude bin` genutzt werden. UVM-SystemC unterstützt nicht direkt das Zusammenfassen mehrerer Variablen in einem Coverpunkt. Des Weiteren kann in UVM-SystemC von zwei verschiedenen Cross-Coverpunkten keine weitere Cross-Coverage gebildet werden (6 Punkte). Dies funktioniert nur bei UVM-Specman *e* und UVM-SystemVerilog (jeweils 10 Punkte).

Checks:
Checks können nur bei UVM-Specman *e* verwendet werden. Damit erlangte UVM-Specman *e* hier die höchste Punktzahl (10 Punkte). UVM-SystemVerilog und UVM-SystemC können Checks nur mit Hilfe von Assertions ausführen. Jedoch können sie in der Regression angezeigt werden. Die Nachbildung über Coverpunkte benötigt mehrere Zeilen im Skript und nimmt deshalb mehr Zeit in Anspruch (je 5 Punkte).

Lines of Code:
Dazu wurden die Lines of Codes der jeweiligen Simulatorskripte der jeweiligen DUVs verglichen. Es fiel auf, dass im SFIFO UVM-Specman *e* die wenigsten Zeilen benötigt. Beim Arbiter jedoch werden in UVM-SystemC die wenigsten Zeilen benötigt. Im SFIFO gibt es bei UVM-SystemC und UVM-SystemVerilog den Nachteil, dass die Bildung von Checks mehr Zeilen im Vergleich zu UVM-Specman *e* benötigt. Andererseits benötigte die Bildung der Coverage in UVM-SystemC mehr Zeilen als bei UVM-SystemVerilog bzw. UVM-Specman *e*. Beim Arbiter fiel auf, dass es möglich ist bei UVM-SystemC Arrays für die Listen zur verwenden. Dies spart bei der Deklaration und beim Aufruf und Speichern der Liste Zeilen ein. Zusammenfassend wurden UVM-Specman *e* und UVM-SystemC mit 8 Punkten, UVM-SystemVerilog mit 5 Punkten bewertet.

UVM-Methoden:
Im Bezug auf die UVM-Methoden können alle 3 Implementierungen alle Eigenschaften und Methoden unterstützen. Bei UVM-SystemC muss der Anwender die UVM-Bibliothek von Accellera selbst einbinden. Würde hingegen die UVM-SystemC Bibliothek der Firma Cadence verwendet müsste der Anwender auf Sequenzen verzichten. Daher wurden für alle UVM-Implementierungen die vollen 10 Punkte vergeben.

Templates:
Bei den Templates gibt es die umfangreichste Unterstützung bei UVM-Specman *e* (9 Punkte). Dabei kann zwischen Interface und Module ENV ausgewählt werden. Für UVM-SystemVerilog gibt es frei verfügbare Templates für die meistbenötigten Module, wie ENV, Agent, Driver usw.. Teilweise mussten diese angepasst werden, um dort einen besseren Nutzen zu erhalten (9 Punkte). Für UVM-SystemC wurden im Zuge dieser Arbeit Templates erstellt. Diese enthielten wie bei UVM-SystemVerilog nur die meistbenötigten Module (5 Punkte).

Wiederverwendung (Re-use):
Bei der Wiederverwendung von Komponenten bieten durch den UVM-Standard alle UVM-Implementierungen die selben Funktionen. Die meisten Funktionen können für alle drei UVM-Implementierungen genutzt werden. Einzelne Unterschiede ergaben sich bei Einbindung von neuen Signalen. Hier mussten bei UVM-Specman *e* die entsprechenden Signal Maps in mehreren Dateien angepasst werden (7 Punkte). Bei UVM-SystemVerilog und UVM-SystemC konnte diese Anpassung im Hauptmodul erfolgen (8 Punkte). Die Änderung im Item war hingegen bei allen UVM-Implementierungen gleich. Im Bezug auf die

Randomisierung fanden sich weitere Unterschiede. So musste bei UVM-SystemC die Randomisierung für das neue Signal in der SEQ-LIB hinzugefügt werden. Bei der Übernahme ganzer Module verhielten sich die UVM-Implementierungen wiederum gleich.

Verification IPs:
Bei den VIPs handelt es sich bei allen Herstellern bspw. um Interfaces wie z.B. I2C oder SPI. Für UVM-SystemVerilog wird ein breites Angebot von VIPs von den Herstellern Cadence [15], Siemens EDA [16] und Synopsys [17] bereitgestellt (10 Punkte). Für UVM-Specman *e* gibt es nur von der Firma Cadence Unterstützung mit einem breiten Abdeckungsbereich (6 Punkte). Dabei werden alle in UVM-SystemVerilog verfügbaren VIPs auch für UVM-Specman *e* angeboten. Für UVM-SystemC können die Lösungen von GreenSocs [18] verwendet werden. Zudem gibt es für einige VIPs von der Firma Cadence die Möglichkeit die VIPs aus UVM-SystemVerilog oder UVM-Specman *e* über UVM-ML einzubinden. Deshalb wurden für UVM-SystemC 4 Punkte vergeben.

Weitergabe an den Kunden:
Für die Weitergabe an Entwicklungspartner und Kunden gibt es verschiedene Voraussetzungen. Diese sind davon abhängig, welche Gegebenheiten dort vorliegen. Falls der Kunde über keine kommerziellen Tools verfügt, ist die Weitergabe bei UVM-SystemC am einfachsten möglich (8 Punkte). Bei UVM-SystemC kann die Verifikation unabhängig von kommerziellen Simulatoren durchgeführt werden. Jedoch ist der Funktionsumfang im Vergleich zu kommerziellen Tools eingeschränkt, z.B. darf das DUV nur in der Sprache UVM-SystemC implementiert sein. Für die Darstellung von Waveforms kann die UVM-SystemC Funktion `sc_trace` benutzt und mit einem Open-Source-Tool zur Anzeige gebracht werden. Als Open-Source-Tool kann beispielsweise `GTKWave` [19] und für die Coverage dann die `FC4SC` Bibliothek verwendet werden. Damit sich der Kunde nicht um die Einbindung der Tools bzw. Bibliotheken kümmern müss, können diese mit Hilfe eines Skripts installiert und initialisiert werden.
Für UVM-Specman *e* ist die Weitergabe nicht ohne weiteres möglich, da hier Lizenzen von Cadence benötigt werden. D.h. bei UVM-Specman *e* ist es von Vorteil, wenn der Kunde bereits UVM-Specman *e* verwendt. Verwendet der Kunde kein UVM-Specman *e* bzw. hat keine entsprechenden Lizenzen, könnte die Weitergabe nur mithilfe spezieller Zugänge zu den Servern des Unternehmens genutzt werden (3 Punkte). Für UVM-SystemVerilog wäre das Vorgehen ähnlich zu UVM-Specman *e*, jedoch gibt es hier durch die Entwicklung von Accellera theoretisch frei verfügbare Compiler wie z.B. Icarus Verilog [20]. Diese Option wurde allerdings im Rahmen dieser Arbeit nicht praktisch untersucht. Darüber hinaus ist es theoretisch möglich UVM-SystemVerilog mit anderen Simulatoren von bspw. Siemens EDA oder Synopsys zu simulieren (7 Punkte).

EDA-Tool-Support:
Der EDA-Tool-Support für die UVM-Implementierungen UVM-Specman *e* (9 Punkte) und UVM-SystemVerilog (10 Punkte) ist sehr ausgeprägt. Für UVM-SystemVerilog gibt es zusätzlich zu den Tools der Firma Cadence Support durch die Hersteller Synopsys und Siemens EDA. Des Weiteren liefert die Anzeige von Fehlermeldungen bei UVM-Specman *e* die präzisesten Informationen. Für UVM-SystemVerilog und UVM-SystemC gibt es theoretisch Support von Siemens EDA und Synopsys. Jedoch fehlen für UVM-SystemC die UVM- und die Coverage Funktionen (3 Punkte).

Code Debugging im EDA-Tool:
Bezüglich des Code Debugging im EDA-Tool lieferte im Rahmen dieser Untersuchung die Firma Cadence für UVM-Specman *e* die ausführlichsten Funktionen, hinzu kamen genaue Fehlermeldungen und die Möglichkeit die Konsole mit der Waveform zu verknüpfen (8 Punkte). Für UVM-SystemVerilog lieferte Cadence ebenfalls ausführliche Funktionen, wie bei UVM-Specman *e*. Einzig bei den Fehlermeldungen gab es teilweise ungenaue Hinweise (7 Punkte). Für UVM-SystemC wird der geringste Funktionsumfang bereitgestellt. Es gibt z.B. keine Möglichkeit die Konsole mit der Waveform zu verknüpfen. Für die Debugging-Funktionen von UVM-SystemC kann der User den GCC Compiler verwenden und er erhält wie bei UVM-Specman *e* und UVM-SystemVerilog ausführliche Debugging-Informationen (7 Punkte).

UVM-Debugging:
Aus der Verifikation des DUV SFIFO und des DUV Arbiter ergab sich, dass von den Tools der Firma Cadence die UVM-Debugging Funktionen nur für UVM-Specman *e* und UVM-SystemVerilog unterstützt werden (jeweils 10 Punkte). Dort war es möglich einzelne Phasen zu simulieren oder nach bestimmten Phasen die Simulation anzuhalten. Des Weiteren war es möglich die verschiedenen Sequenzen mit den jeweiligen Werten anzuzeigen. In den Tools der Firma Cadence gibt es derzeit keine Unterstützung für UVM-SystemC (0 Punkte).

Anzeigen und Ausblenden von Konsolenausgaben:
Für das Anzeigen und Ausblenden von Konsolenausgaben gibt es für jede UVM-Implementierung Möglichkeiten die Anzeigen zu filtern. Beispielsweise ist es möglich, nur Meldungen mit einer bestimmten

Tabelle IV
SIMULATIONSZEIT DES SFIFO.

Durchlauf	Simulationszeit in s		
	UVM *e*	UVM SV	UVM SC
1	3,3	1,5	30,5
2	3,2	1,0	32,9
3	3,6	2,6	29,9
Durchschnitt	3,4	1,7	31,1

Tabelle V
SIMULATIONSZEIT DES ARBITER.

Durchlauf	Simulationszeit in s		
	UVM *e*	UVM SV	UVM SC
1	4,2	3,0	58,8
2	2,7	2,3	40,6
3	2,6	2,7	59,0
Durchschnitt	3,2	2,7	52,8

Wertigkeit anzuzeigen. Zudem kann festgelegt werden, ob bei einem illegalen Coverpoint oder UVM-Error die Simulation anhalten oder weiterlaufen soll. Hier ist es auch möglich, bei illegalen Bins die Simulation zu stoppen oder weiterlaufen zu lassen (je 10 Punkte).

Lizenzkosten:
Lizenzkosten fallen für die Tools von Cadence an, deshalb wurde für UVM-Specman *e* 1 Punkt vergeben. UVM-SystemVerilog wurde in dieser Arbeit mit dem Tool Xcellium der Firma Cadence simuliert. Jedoch ist es theoretisch möglich UVM-SystemVerilog mit frei verfügbaren Tools zu simulieren. Aus diesem Grund wurden 3 Punkte vergeben. UVM-SystemC kann ohne kommerzielle Simulatoren simuliert werden, jedoch muss das DUV dann auch in UVM-SystemC implementiert sein. Wird UVM-SystemC bspw. mit dem Tool Xcellium der Firma Cadence simuliert, fallen Lizenzkosten an (6 Punkte). Alternativ kann UVM-SystemC mit einem frei verfügbaren Simulator verknüpft werden, um Designs in VHDL oder Verilog einzubinden. Jedoch erfordert dies Aufwand und wurde in dieser Arbeit nicht geprüft.

Simulationszeit
Die Simulationszeit des SFIFOs ist in Tabelle IV, die Simulationszeit des Arbiters in Tabelle V dargestellt. Bei beiden Modulen ist die Simulationszeit bei der UVM-Implementierung in UVM-SystemVerilog am kürzesten. Daher wurde UVM-SystemVerilog mit 10 Punkten bewertet. UVM-Specman-*e* benötigt bei beiden Modulen eine etwas höhere Simulationszeit als UVM-SystemVerilog und wird daher mit 7 Punkten bewertet. UVM-SystemC hat bei beiden Modulen die - verglichen mit UVM-SystemVerilog bis zum Faktor 20 - mit Abstand höchste Simulationszeit und erhält daher 1 Punkt.

UVM-Multi-Language:
Die Verifikation von DUVs, die in unterschiedlichen HDLs modelliert sind, wird von den Tools der Firma Cadence für alle drei UVM-Implementierungen unterstützt. Für UVM-SystemVerilog und UVM-Specman *e* kann dies ohne Einschränkungen verwendet werden. Daher wurden für diese beiden UVM-Implementierungen jeweils 10 Punkte vergeben. Für UVM-SystemC ist es auch möglich diese Methodik anzuwenden. Hierbei gibt es allerdings die Einschränkung, dass Xcelium keine Sequenzen unterstützt. Daher konnten diese nicht mit eingebunden werden. Aufgrund der fehlenden Unterstützung von Sequenzen in UVM-SystemC wurden nur 5 Punkte vergeben.

Regression:
Bei den Regression-Tools wurden die größten Unterschiede zwischen UVM-SystemC, UVM-SystemVerilog und UVM-Specman *e* festgestellt. Für alle drei UVM-Implementierungen kann angezeigt werden, welche Testcases ohne und welche mit einem Fehler durchgelaufen sind. Zudem ist es möglich für DUVs in VHDL oder Verilog die Coverage anzeigen zu lassen. Für die Coverage und Checks innerhalb der Verifikation ist es nur für UVM-Specman *e* und UVM-SystemVerilog möglich diese zur Anzeige zu bringen. Hierbei kann bei UVM-Specman *e* zusätzlich zwischen Coverage und Checks unterschieden werden. Für UVM-SystemC muss die separate `FC4SC` Bibliothek verwendet werden.
Beim Aufsetzen der Tool-Umgebung verhielten sich die drei UVM-Implementierung ähnlich. Für jede UVM-Implementierung wird ein Skript benötigt, welches das Tool initialisiert. Hierbei können mit einer zusätzlichen Datei vom Nutzer persönliche Einstellungen und Konfigurationen hinzugefügt werden. Bei UVM-Specman *e* und UVM-SystemVerilog werden jedoch die meisten Einstellungen vom Tool selbst übernommen. So müssen z.B. dem `xrun` nur die entsprechenden Befehle und Dateien mitgegeben werden. Bei UVM-SystemC muss jeder Simulator für die entsprechende Sprache selber initialisiert werden und zum Schluss müssen alle Simulationsergebnisse im Hauptsimulator verknüpft werden. Dazu sollte gewährleistet werden, dass alle Ergebnisse im selben Ordner gespeichert wurden bzw. zusätzlich die einzelnen Pfade angegeben werden. Bei der Regression gibt es die beste Unterstützung für UVM-Specman *e* (9 Punkte) und UVM-SystemVerilog (8 Punkte). Für UVM-SystemC fallen diese Regression-Funktionen weg, es kann lediglich angezeigt werden, ob ein Testcase fehlschlägt oder nicht (4 Punkte).

Tabelle VI
BEWERTUNGSTABELLE.

Kategorie	Gesamt-punktzahl (G)	UVM-Specman *e*		UVM-SystemVerilog		UVM-SystemC	
		Bewer-tung (B)	G·B	Bewer-tung (B)	G·B	Bewer-tung (B)	G·B
DUV Language	18	10	180	10	180	5	90
TLM-Kommunikation	25	10	250	10	250	8	200
Konfigurierbarkeit	19	10	190	10	190	10	190
Constrained Randomization	32	9	288	9	288	9	288
Coverage	32	10	320	10	320	6	192
Checks	30	10	300	5	150	5	150
LOC	10	8	80	5	50	8	80
UVM-Methoden	16	10	160	10	160	10	160
Templates	8	9	72	9	72	5	40
Wiederverwendung (Re-use)	24	7	168	8	192	8	192
VIPs	7	6	42	10	70	4	28
Weitergabe an den Kunden	3	3	9	7	21	8	24
EDA-Tool-Support	21	9	189	10	210	3	63
Code Debugging im EDA-Tool	24	8	192	7	168	7	168
UVM-Debugging	21	10	210	10	210	0	0
Anzeigen bzw. Ausbleden von Konsolenausgaben	13	10	130	10	130	10	130
Lizenzkosten	40	1	40	3	120	6	240
Simulationszeit	23	7	161	10	230	1	23
UVM-ML	7	10	70	10	70	5	35
Regression	26	9	234	8	208	4	104
Ein- und Ausblenden von Coverpunkten	21	10	210	10	210	1	21
Gesamtsumme			**3495**		**3499**		**2418**

Ein- und Ausblenden von Coverpunkten: Mithilfe der Tools der Firma Cadence lassen sich nur bei UVM Specman *e* und UVM-SystemVerilog coverage- und check-Punkte anzeigen und im Nachhinein ausblenden (je 10 Punkte). Diese Funktion war beim Arbiter hilfreich, denn es wurden nicht alle Coverpunkte erreicht, da die Fehler nicht korrigiert wurden. Allerdings möchten User angezeigt bekommen, ob die Coverage erreicht wurde oder nicht. Bei UVM-SystemC ist dies nicht möglich, hier müssen die Coverage Punkte im Code ausgeblendet werden (1 Punkt). Das Ausblenden im Code führt dazu, dass die Regression neu gestartet werden muss. Sind die Module komplex, wird dazu viel Zeit benötigt. Alternativ kann die Anzeige in der `FC4SC` Bibliothek nach vier Kategorien gefiltert werden, die Gesamtwertung wird jedoch weiterhin von allen Coverpunkten gebildet. Bei kleineren DUVs wie dem SFIFO oder dem Arbiter ist dies ein geringer Aufwand. Für erfahrene Nutzer können jedoch in der XML-Datei die entsprechenden Coverpunkte ausgeblendet werden. Dadurch kann dieselbe Flexibilität erreicht werden, wie bei der Regression in Tools der Firma Cadence.

IV. BEWERTUNG DER UVM-IMPLEMENTIERUNGEN

Nach Abschluss der Bewertung aller 21 Kriterien wurde die jeweilige Bewertung (B) jedes einzelnen Kriteriums entsprechend Kapitel III-F mit der Gesamtpunktzahl (G) jedes Kriteriums aus Tabelle II für jede der drei UVM-Implementierungen multipliziert. Die Gesamtpunktzahl jedes Kriteriums aus Tabelle II ist in Tabelle VI in der zweiten Spalte dargestellt. In den Spalten 3, 5 und 7 findet sich die jeweilige individuelle Bewertung des Kriterums, wie sie in Kapitel III-F erläutert wurde. Die Produkte aus Gesamtpunktzahl und individueller Bewertung (G·B) sind in den Spalten 4, 6 und 8 dargestellt. Durch Aufsummieren der Produkte aus Gesamtpunktzahl und indiviueller Bewertung ergibt sich die Gesamtsumme pro UVM-Implementierung. Diese ist in der untersten Zeile von Tabelle VI dargestellt und gibt Aufschluss darüber welche UVM-Implementierung am besten geeignet ist. UVM-SystemVerilog erreicht mit 3499 Punkten die höchste Punktzahl. UVM-Specman *e* hat mit 3495 Punkten die zweithöchste Punktzahl und UVM-SystemC mit 2418 Punkten die geringste Punktzahl.
UVM-SystemVerilog und UVM-Specman *e* sind in vielen Kategorien ähnlich bewertet, daher beträgt der Unterschied zwischen den beiden UVM-Implementierungen nur 4 Punkte. Gemeinsamkeiten gibt es beispielsweise bei der DUV-Language oder der TLM-Kommunikation. Die größten Unterschiede liegen in den Lizenzkosten, den Checks und den VIPs.
UVM-SystemC verliert in den Kategorien EDA-Tool-Support, UVM-Debugging und Ein- und Ausblenden von Coverpunkten deutlich gegenüber den anderen beiden UVM-Implementierungen. Dies bedeutet, dass der Unterschied hauptsächlich an den Tools der EDA-

Hersteller liegt. Die größten Vorteile erreicht UVM-SystemC bei der Konfigurierbarkeit und den Lizenzkosten. Zudem liegt UVM-SystemC bei den Lines of Code gleichauf mit UVM-Specman *e*. Bei der Wiederverwendung gibt es zudem kaum Unterschiede zwischen UVM-SystemVerilog und UVM-SystemC.

V. Zusammenfassung

Im Rahmen dieser Arbeit wurde gezeigt, dass UVM-SystemC aktuell für die Schaltungsverifikation anwendbar ist. UVM-SystemC unterstützt alle UVM-Methoden und es ist möglich Designfehler zu identifizieren. Es kann eine Coverage gebildet und durch das Seed Management können verschiedene Seeds verwendet werden. Weiterhin können neben DUVs in UVM-SystemC auch DUVs verifiziert werden, die in den HDLs VHDL oder Verilog modelliert sind. Durch die Unterstützung von UVM-ML ist es möglich, Designs, die aus Blöcken in unterschiedlichen HDLs bestehen, zusammen zu verifizieren.

Die größten Vorteile der Schaltungsverifikation mit UVM-SystemC liegen in den nicht vorhandenen Lizenzkosten. UVM-SystemC kann kostenlos genutzt werden. Das erleichtert auch die Zusammenarbeit mit Entwicklungspartnern, Kunden oder externen Designhäusern, insbesondere wenn diese keine kommerziellen CAD-Tools lizensiert haben.

Bei den Kriterien Konfigurierbarkeit, Constrained Randomization, den benötigten Lines of Code für das Verifikationsskript, UVM-Methoden und Re-Use ist UVM-SystemC den beiden etablierten UVM-Implementierungen durchaus ebenbürtig, teilweise sogar überlegen.

Jedoch gibt es für UVM-SystemC Einschränkungen gegenüber den beiden UVM-Implementierungen UVM-Specmen *e* und UVM-SystemVerilog.

Großes Verbesserungspotential besteht beim eingeschränkten Support der Fa. Cadence für UVM-SystemC. Der Wegfall der UVM-Debugging Möglichkeiten, ist hier der größte Nachteil. Hierbei wäre es wünschenswert, wenn die Firma Cadence die vollständige UVM-Bibliothek bereitstellen würde. Der Anwender kann die UVM-Bibliothek selber einbinden, muss sich jedoch bei der Ausführung der Simulation um viele Einstellungen selbst kümmern.

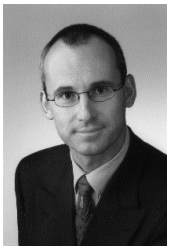
Eine weitere Möglichkeit zur Verbesserung liegt beim Simulator. Hier sollte die Möglichkeit bestehen zwischen der Konsole und der Waveform zu wechseln. Für das Anzeigen der Coverage wurde bis zum Ende dieser Arbeit keine Möglichkeit gefunden, sie in der Regression anzeigen zu können. Jedoch ist es möglich die Coverage mithilfe der `FC4SC` Bibilothek zur Anzeige zu bringen. Dabei muss der User jedoch Einschränkungen beim Ein- und Ausblenden von Coverpunkten hinnehmen.

Literaturverzeichnis

[1] Accellera Systems Initiative, ”Universal Verification Methodology (UVM) 1.2 User’s Guide”, https://www.accellera.org/downloads/standards/uvm, 21.10.2022.
[2] Accellera Systems Initiative, ”UVM-SystemC Library 1.0-beta2”, https://www.accellera.org/downloads/drafts-review, 21.10.2022.
[3] Cadence Design Systems, ”Universal Verification Methodology (UVM) e User Guide”, 2014.
[4] Accellera Systems Initiative, ”SystemC Verification 2.0.1: SystemC Verification Library”, https://www.accellera.org/downloads/standards/systemc, 21.10.2022.
[5] Group of Computer Architecture of the University of Bremen FB3, ”CRAVE: Constrained Random Verification Environment”, http://www.informatik.uni-bremen.de/agra/systemc-verification/crave.html, 21.10.2022.
[6] Dragos Dospinescu, ”FC4SC” https://www.amiq.com/consulting/2018/11/15/new-release-of-the-functional-coverage-for-systemc-library/, 21.10.2022.
[7] Cadence Design Systems, ”SystemC SCV/CVE Library Reference: Product Version 14.1”, 2014.
[8] AlDanial, ”cloc”, https://github.com/AlDanial/cloc/releases/tag/1.84, 21.10.2022.
[9] AMIQ EDA, ”DVT Eclipse IDE”, https://dvteclipse.com/products/dvt-eclipse-ide, 21.10.2022.
[10] Cadence Design Systems, ”Xcelium: 18.09.007,” https://www.cadence.com/ko_KR/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html, 14.03.2020.
[11] Accellera Systems Initiative, ”SystemC 2.3.3 (Includes TLM): Core SystemC Language and Examples”, https://www.accellera.org/downloads/standards/systemc, 21.10.2022.
[12] Cadence Design Systems, ”UVM-SC Library Reference: UVM Version 1.0”, 2011.
[13] Mentor, a. Siemens Business, ”Questa Verification Solution”, https://eda.sw.siemens.com/en-US/ic/questa/ 21.10.2022.
[14] Synopsys Inc., ”Verdi”, https://www.synopsys.com/verification/debug/verdi.html, 21.10.2022.
[15] Cadence Design Systems, ”Verification IP”, https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html, 21.10.2022.
[16] Siemens EDA, ”Verification IP”, https://eda.sw.siemens.com/en-US/ic/questa/simulation/verification-management/, 21.10.2022.
[17] Synopsys Inc., ”Verification IP”, https://www.synopsys.com/verification/verification-ip.html, 21.10.2022.
[18] Mark Burton, ”GreenSocs”, https://www.machineware.de/#qemu, 21.10.2022.
[19] Udi Finkelstein, ”GTKWave,” http://gtkwave.sourceforge.net/, 21.10.2022.
[20] Stephen Williams, ”Icarus Verilog”, http://iverilog.icarus.com/home, 21.10.2022.

Moritz Kupke schloss sein Bachelorstudium der Elektrotechnik und Informationstechnik mit der Vertiefungsrichtung Mikroelektronik an der Hochschule Düsseldorf im Jahr 2017 ab. Seine Masterstudium Mikroelektronik - wiederum an der Hochschule Düsseldorf - schloss er im Jahr 2020 ab. Parallel zu seinem Masterstudium war er von 2017 bis 2019 wissenschaftlicher Mitarbeiter an der Hochschule Düsseldorf. Seit Mitte 2020 ist er als ASIC-Entwicklungsingenieur bei der Firma Bosch GmbH in Dresden beschäftigt.

Stephan Gerth schloss sein Studium der Informatik an der Brandenburgischen Technischen Universität Cottbus (BTU) im Jahr 2008 ab. Im Anschluss arbeitete er im Fraunhofer Institut IIS/EAS als wissenschaftlicher Mitarbeiter am Entwurf von ESL-Modellen und der dazugehörigen Entwurfs- und Verifikationsmethodik und übernahm später die Gruppenleitung. Seit 2018 ist er für die Robert Bosch GmbH am Standort Dresden tätig und leitet die Verifikation in verschiedenen Projekten.



Bernhard Rieß schloss sein Studium der Elektrotechnik und Informationstechnik an der Technischen Universität München im Jahr 1992 ab. Anschließend war er wissenschaftlicher Mitarbeiter am Lehrstuhl für Rechnergestütztes Entwerfen an der TU München und wurde dort 1996 zum Dr.-Ing. promoviert. Seit Anfang 1997 war er als Entwicklungsingenieur bei der Infineon Technologies AG in München/Neubiberg tätig. 2012 wurde er von der Hochschule Düsseldorf als Professor für das Lehrgebiet Mikroelektronik berufen.

MULTI PROJEKT CHIP GRUPPE

Hochschule Aalen
Prof. Dr. Bürkle, (07361) 576-2103
heinz-peter.buerkle@htw-aalen.de

Hochschule Albstadt-Sigmaringen
Prof. Dr. Gerlach, (07571) 732-9155
gerlach@hs-albsig.de

Hochschule Esslingen
Prof. Dr. Lindermeir, (0711) 397-4221
walter.lindermeir@hs-esslingen.de

Hochschule Furtwangen
Prof. Dr. Benyoucef, (07723) 920-2342
bed@hs-furtwangen.de

Hochschule Heilbronn
Prof. Dr. Gessler, (07940) 1306-184
gessler@hs-heilbronn.de

Hochschule Karlsruhe
Prof. Dr. Ng, (0721) 925-1520
Herman-Jalli.Ng@hs-karlsruhe.de

Hochschule Konstanz
Prof. Dr. Schick, (07531) 206-657
cschick@htwg-konstanz.de

Hochschule Mannheim
Prof. Dr. Giehl, (0621) 292-6860
j.giehl@hs-mannheim.de

Hochschule Offenburg
Prof. Dr. Mackensen, (0781) 205-4770
elke.mackensen@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. Hennig, (7121) 271-7129
eckhard.hennig@reutlingen-university.de

Technische Hochschule Ulm
Prof. Dr. Terzis, (0731) 96537-627
anestis.terzis@thu.de

www.mpc-gruppe.de

© 2024 MPC-Gruppe
Das Werk und seine Teile sind urheberrechtlich geschützt. Jede Verwertung in anderen als den gesetzlich zugelassenen Fällen bedarf deshalb der vorherigen schriftlichen Einwilligung des Herausgebers Prof. Dr. Lothar Schmidt, MPC-Gruppe, Albert-Einstein-Allee 53, D-89081 Ulm.