Design Method for Multiplier-Less Two-Variable Numeric Function Approximation

Jochen Rust and Steffen Paul
Institute of Electrodynamics and Microelectronics (ITEM.me)
University of Bremen, Bremen, Germany, +49(0)421/218-62538
Email: {rust, steffen.paul}@me.uni-bremen.de

***Abstract*—In this paper a novel method for hardware-based realization of two-variable numeric functions is introduced. The main idea is based on the extension of the well-known piecewise linear approximation technique, which is often used for the calculation of one-variable elementary functions. A non-uniform and plane segmentation scheme enables quick segment access at runtime; the use of multiplier-less linear equations causes high performance in terms of throughput. As both the extraction of approximation-related parameters and its mapping to corresponding hardware elements is automated, the design time is also reduced to a minimum.**

For evaluation, several approximations with varying constraints are generated and compared on the algorithmic level to one another as well as to actual references. In conjunction with the results of logical and physical CMOS synthesis, our work turns out to be highly efficient in terms of throughput, memory requirements and energy consumption.

***Index Terms*—numeric function approximation, two-variable, multiplier-less**

I. Introduction and Related Work

Efficient signal processing of numeric functions has been a demanding challenge for decades [1]. A large number of existing approaches take a vast range of different mathematical techniques into account [2]. In the scope of application-specific signal processing, e.g., those considered for the design of application-specific integrated circuits (ASIC), the efficient realization of elementary functions has always been an important topic [3] [4]. Thus, it has been exhaustively examined in various application-areas, e.g., neuronal networks [5] or computer graphics [6]. However, for the efficient hardware-mapping of two-variable (also called two-dimensional (2D)) numeric functions, there exists only a small set of architectures, mostly adapted directly to a specific application (for example, see [7]).

The hardware-mapping of one-variable elementary functions is mainly based on ROM techniques, iterative methods, function graph approximation or a combination of these aspects [2]. In most cases, generic approaches for 2D numeric functions refer to the function approximation method [8][9], as ROM techniques may lead to impractical high memory requirements, even if compression techniques are considered. Although some iterative methods are able to process 2D functions, e.g., CORDIC [2], this ability is limited to a set of fixed functions just as the (approximation) accuracy is directly connected to the number of iteration steps.

A practical design method for a hardware-convenient 2D numeric function approximation (NFA) using a piecewise polynomial approximation scheme is presented by Nagayama et al. [9]. There, bilinear interpolation polynomials and a planar segmentation scheme are combined in order to achieve high accuracy NFAs. However, both the proposed approximation method and the segment encoding technique lead to a comparatively high signal processing effort considering nowadays one-dimensional function approximation approaches. Also, the handling of symmetric functions ($f(x_1, x_2) = f(x_2, x_1)$) is rather difficult: As this type of function normally allows to swap the input operands, this may be used in order to decrease the range of values. In the scope of piecewise NFA, this can be exploited in order to reduce the segmentation effort nearly by half. However, using bilinear interpolation polynomials, additional control effort is required as the coefficients must be reassigned [9].

In this paper, the design method of the high-performance automatic linear function approximation (ALFA) approach for one-dimensional functions, as described in [10], is extended to 2D numeric functions, comprising the parameter extraction of appropriate linear approximations and its mapping to corresponding hardware elements. In detail, our proposed work covers the following features:

- An automated approach for piecewise linear function approximation of 2D NFAs is presented that creates a HDL-based hardware description of a given function by the specification of an average (ε_{avg}) or a maximum (ε_{max}) error. Thus, this work can be seen as a high-level synthesis step for functions with two variables in the digital design flow.
- In order to achieve high-speed signal processing, multiplier-less NFAs are generated reducing the calculation effort to a minimum. For quick segment selection at runtime, a hardware-convenient, non-uniform and planar segmentation scheme is used.
- Besides the HDL-based hardware description, the proposed work also provides a testbench for hardware verification as well as a MATLAB-model enabling simulation on the algorithmic level.

II. Preliminaries

Definition 1: A **segment** denotes a specified (sub-)area of a given 2D function. In this paper a segment is determined by

a start and end point argument (see (2)).

Definition 2: An **error** determines the difference between the (quantized) absolute values of the original function and its NFA. A (local) **average error** (ε_{avg}) is the arithmetic mean of all error values inside a segment. A **maximum error** (ε_{max}) is the highest error inside a segment. The **specified error** is the user-defined constraint determining the requested approximation quality considering ε_{max} or ε_{avg}

Definition 3: The **quantization factor** (QF) denotes the overall number of partial products considered for multiplier-less coefficient estimation.

Definition 4: The **resolution** X denotes the smallest considerable value that in occurs in the chosen fixed point format due to digital quantization effects

III. Multiplier-less 2D NFA

The fundamental concept behind piecewise function approximation refers to the function graph reconstruction by simplified means [2]. In order to reduce the signal processing effort, a partial gradient based approach (see Sec. III-A) is used. This causes a simplification of the calculation effort to the bilinear equation

$$\tilde{f}(x_1, x_2) = \mathbf{x}\tilde{\boldsymbol{\beta}} = \tilde{\beta}_0 + \tilde{\beta}_1 x_1 + \tilde{\beta}_2 x_2 \ , \tag{1}$$

with x_d as the two input variables, $d \in \{1, 2\}$ as the dimension index, $\boldsymbol{\beta}$ as the linear coefficients and $\tilde{f}(x_1, x_2)$ as the resulting NFA. For non-uniform segmentation, the original function $f(x_1, x_2)$ is divided into several sub-functions with variable input ranges (see Fig. 1). In order to enable quick access to these sub-functions, a restricted non-uniform segmentation scheme is applied where the range and the size of each segment is defined by

$$s_d(k_d) \leq x_d < s_d(k_d + 1) \ , \tag{2}$$

with

$$s_d(k_d) = \begin{cases} s_d(k_d - 1) + [\mathrm{B}_d - \mathrm{A}_d] \cdot 2^{-h_{k_d}} & , k_d > 0 \\ \mathrm{A}_d & , \text{else} \end{cases} , \tag{3}$$

with A_d, B_d as the start and end points of $f(x_1, x_2)$, k_d as segment function indexes determining the start and end points of sub-segments and $h_{k_d} \in \mathbb{N}^+$ as the interval exponents of the k_d-th segment. Here, only input operands with an equal start and end point are regarded which leads to $\mathrm{A}_d = \mathrm{A}$ and $\mathrm{B}_d = \mathrm{B}$.

Constraining the overall function range to

$$\mathrm{B} - \mathrm{A} = 2^{\mathrm{H}} \ , \tag{4}$$

with $\mathrm{H} \in \mathbb{Z}$, hardware-convenient access of all segments is permitted, considering the most significant bits (MSB) of the input values. As h_{k_d} may vary for each single segment and, consequently, also the number of affected MSBs (see Fig. 1), a non-uniform segmentation scheme is obtained.

In order to achieve the requested multiplier-less signal processing, further restrictions must be formulated. Thus, the linear coefficients are realized by a limited set of partial products [11]. The corresponding calculation is reduced to a small number of shift and add operations, determined by q =QF.

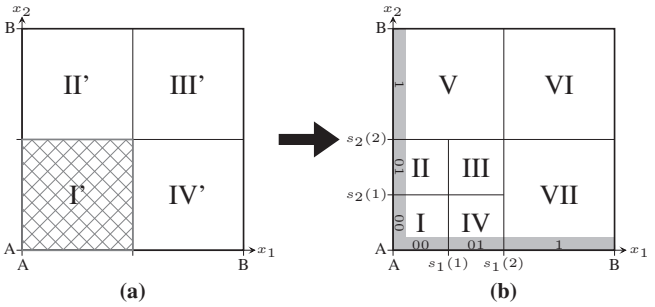

Fig. 1. Example of the proposed non-uniform planar segmentation scheme. The gray grid in (**a**) depicts a segment which is refined by the heuristic described in Sec. III-A. (**b**) shows the arising segmentation with corresponding MSB assignment and start and end point determination (see (2)).

Considering all mentioned aspects, the function gradients in (1) can be expressed as

$$\tilde{\beta}_{d,q} = \sum_{j=1}^{q} \pm \left(2^{\lambda_{d,j}}\right) \ , \tag{5}$$

where $\lambda_{d,j}$ denotes the exponent of the j-th partial product. The equation for the proposed multiplier-less 2D NFA is given by

$$\tilde{f}(x_1, x_2) = \mathbf{x} \begin{pmatrix} \tilde{\beta}_{01} & \tilde{\beta}_{02} & & \tilde{\beta}_{0m} \\ \tilde{\beta}_{11,q} & \tilde{\beta}_{12,q} & \dots & \tilde{\beta}_{1m,q} \\ \tilde{\beta}_{21,q} & \tilde{\beta}_{22,q} & & \tilde{\beta}_{2m,q} \end{pmatrix} \cdot \boldsymbol{\kappa}(\mathbf{x}) \ , \tag{6}$$

with m as the number of segments in total, $\mathbf{x} = (1 \ \ x_1 \ \ x_2)$ and $\boldsymbol{\kappa}(\mathbf{x})$ as fade-out function realizing the selection of the actual segment according to the definition in (2).

A. Parameter Extraction

For the extraction of function parameters which are required in order to set up an appropriate NFA, several hardware constraints have to be declared. In detail, ε_{max} or ε_{avg}, the QF, A, B, the data path resolution and the input port size must be specified in advance.

For an automated 2D NFA generation of $f(x_1, x_2)$, we propose a distinctive heuristic that can be seen as the extension of the one-dimensional ALFA NFA [10]. In a first step, linear coefficients are estimated which are used as a reference for the resulting multiplier-less approximation. In detail, this is done by calculating the arithmetic mean of partial gradients. Considering digital quantization effects ($x_d = n_d X$), the (linear) reference coefficients of the original function can be calculated

$$\begin{pmatrix} \beta_1' \\ \beta_2' \end{pmatrix} = 2^{-2\mathrm{H}} \cdot \left[\sum_{n_1=0}^{2^{\mathrm{H}}-1} \left[\sum_{n_2=0}^{2^{\mathrm{H}}-1} \boldsymbol{\nabla} f(n_1 X, n_2 X) \right] \right] \ , \tag{7}$$

with $\boldsymbol{\nabla} = \begin{pmatrix} \frac{\partial}{\partial n_1} & \frac{\partial}{\partial n_2} \end{pmatrix}^T$. For the estimation of partial gradients in sub-segments, $f(n_1 X, n_2 X)$ and H must be adapted to the corresponding parameters.

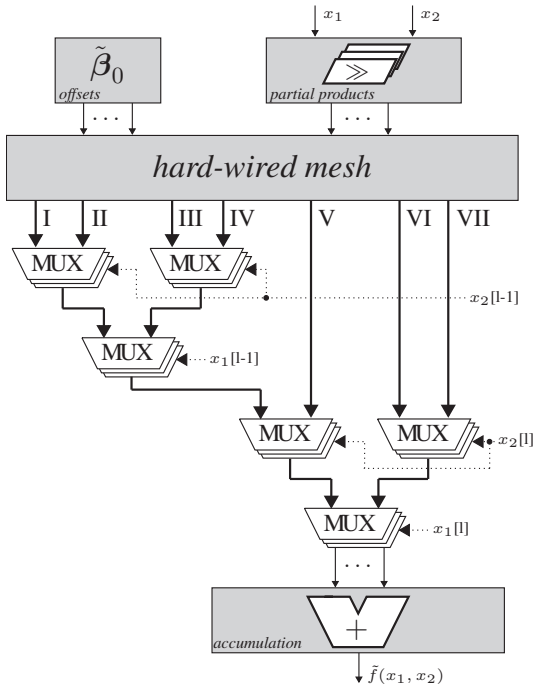

Fig. 2. Multiplier-less hardware architecture of the corresponding 2D NFA example from Fig. 1 with the offsets $\tilde{\boldsymbol{\beta}}_0$, the QF-constrained partial products as input values and the accumulating (tree-adder) unit at the output. l denotes the bit width of the input operands. The segmentation is realized by fixed wired multiplexer units. All values that belong to the same segment and single values are denoted by thick and thin arrows, respectively. Control data is marked by the dotted lines.

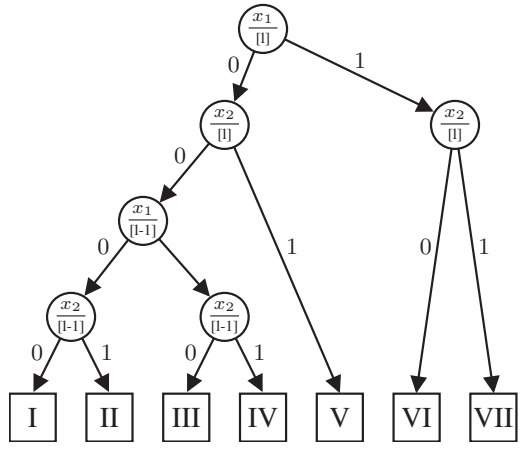

Fig. 3. Tree-representation of the segmentation example introduced in Fig. 1. The nodes signify the bit that is taken into account for selecting an active segment (expression below the line) of l-bit sized input operands (variable above the line). The edges mark the particular bit values.

The multiplier-less coefficients $\tilde{\beta}_1, \tilde{\beta}_2$ are achieved by iterative superposition of partial products: In each iteration step $\tilde{\beta}_1, \tilde{\beta}_2$ are refined by adding the partial product with the least deviation to β_1' and β_2', respectively. QF represents the number of iteration steps.

The offset $\tilde{\beta}_0$ is determined using an adapted bisection method, halving the set of candidates iteratively by considering the midpoints of the offset range: $\tilde{\beta}_0^{(\gamma)} = \pm\left((\mathrm{B}-\mathrm{A})\cdot 2^{-\gamma}\right)$, with γ as the iterator (, e.g., for $\mathrm{B}-\mathrm{A}=2^8$, the estimation starts with the values $\tilde{\beta}_0^{(1)} = \pm 128$). In order to decide whether the actual offset has to be added or subtracted, ε_{max} or ε_{avg} are taken into account, according to the specified error mode. In detail, the respective error function is calculated in each iteration step and the corresponding offset range with the smaller error is selected. The offset estimation stops when the offset refinement is smaller than the data path resolution ($(\mathrm{B}-\mathrm{A})\cdot 2^{-\gamma} < X$).

After the NFA coefficients have been estimated, the approximation quality is rated taking ε_{max} or ε_{avg} into account. As the corresponding calculation is already performed in the last offset iteration step, no additional calculation effort is required for this. If the NFA does not reach the specified error, the segment is split up into four quadrants with an equal size and the resulting segments are processed sequentially starting at the lower left (see Fig. 1). Otherwise, if the specified error constraint is satisfied, the heuristic moves on to the next segment, processing the remaining segment in a clock-wise manner. By repetition of the refinement step at different sub-function ranges, a set of non-uniform segments is achieved. If four related quadrants have been successfully approximated, the heuristic starts over with the next segment on the superior level, until the entire function range is processed in this manner. For each segment that has been successfully approximated, the parameters $\tilde{\boldsymbol{\beta}}_{d,q}, \tilde{\boldsymbol{\beta}}_0, \boldsymbol{\lambda}_{d,j}, h_{k_d}$ and $s_d(k_d)$ are extracted and memorized.

In order to improve the described heuristic, it features some additional supplements: As mentioned in Sec. I, the segmentation effort can be decreased by half in case of symmetric functions. Hence, at the beginning of the parameter extraction this function type is automatically detected by comparison of all function values with the results achieved by operand swapping. If a symmetric function is recognized, the affected negligible sub-function range will be skipped during parameter extraction. However, the implementation of the operand swapping must be integrated manually by now. Another feature covers the the disregard of of special values (for example infinity) or user-specified sub-function ranges. Here, a separate treatment can be selected that enables a direct assignment of the corresponding function values to the output. Though this usually causes a higher number of constants in total, the affected sub-function range will be ignored during parameter extraction. As this may significantly simplify the approximation effort, the overall number of segments could decrease evidently and, consequently, a higher throughput can be achieved.

B. Hardware mapping

After the parameter extraction is finished, a VHDL-based hardware description of the 2D NFA is created (see Fig. 2). For this purpose, template files providing a generic structure of the demanded architecture are used that base on the StringTemplate code generator [12]. The input and output ports as well as the constants and the result accumulation are

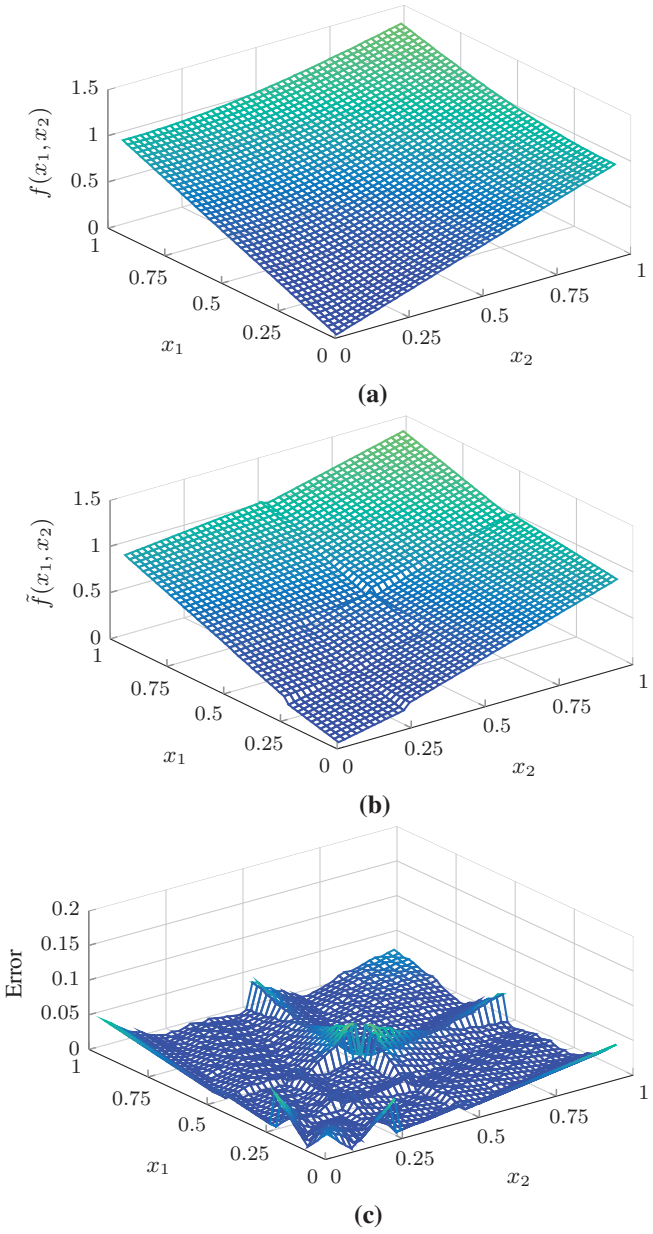

Fig. 4. Overview of the application of the proposed approximation approach to the Euclidean norm function with (**a**) the original function, (**b**) the NFA with the segmentation scheme introduced in Fig. 1 and (**c**) depicting the distribution of errors over the whole function range.

directly mapped to corresponding VHDL-expressions. For the quantized gradients, the input operands are shifted accordingly, which is realizable by common VHDL means. The segmentation is realized using multiplexers that refer to encapsulated if-statements in VHDL. In order to enable a straight-forward StringTemplate processing, the MSB data is transferred to a binary tree datatype that is traversed in a depth-first manner during the code generation phase. An example of a binary tree is given in Fig. 3.

Another important aspect concerning the hardware-mapping of algorithms is the selection of an appropriate number format.

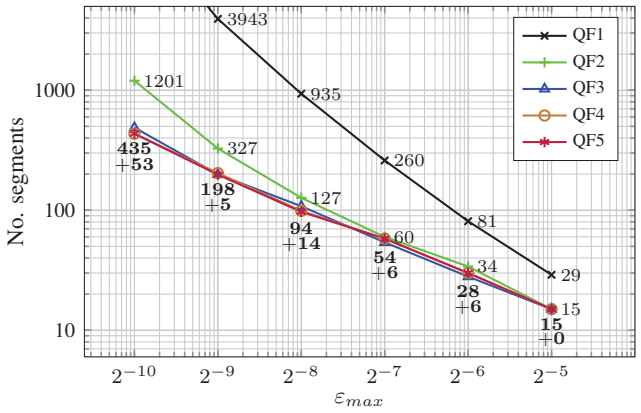

Fig. 5. Segmentation effort over the specified maximum error for approximations of the Euclidean Norm function $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ with varying QFs. Bold expressions mark the number of segments for QF3-QF5 with the added denoting range. Other values depict the number of segments for QF1 and QF2.

For our proposal, the Q-fixed point format is used and adapted to the width of the input operands and the constants as specified in Sec. III-A. The output port is set up considering the highest possible resulting value. By default, the internal data path and the output port have the same bit width. However, if any partial product requires higher values that are not realizable by this setup, the internal data path is extended accordingly.

In order to achieve further signal processing improvement, a reduction of the binary tree is considered: For symmetry functions, some nodes of the binary tree may possess only one single child. As these sub-tree structures do not contribute the localization of segments, these nodes are bypassed. From the hardware point of view this refers to the reduction of obsolete bits of an MSB-sequence. Thus, after the binary tree is created, a tree-reduction is performed by depth-width traverse.

C. Simulation and Verification

Beside the VHDL-based hardware architecture, both a simulation model and a testbench can be additionally generated. As the required parameters are already extracted, this is done again by the use of StringTemplates. For the simulation model, a MATLAB-based function that possesses the same behavior as the hardware architecture is considered, using the fixed-point toolbox. The testbench consists of a VHDL-framework including the instantiation of the NFA design as well as an automated connection of all specified ports and signals. By default, an exhaustive stimulation scheme is provided by the use of two encapsulated loops, where the loop variables are assigned to the input ports.

IV. Results

In order to qualify the proposed NFA approach, several approximations are created, analyzed and compared to actual references. Additionally, the evaluations are performed considering the approximation quality, algorithmic and hardware performance. Further, the function range is set to $0 \leq x_d < 1$

TABLE I
ALGORITHMIC PERFORMANCE OF THE SELECTED APPROXIMATIONS FOR THE EUCLIDEAN NORM FUNCTION COMPARED TO ACTUAL REFERENCES.

	This work		[13]	[9]
Technique	Multiplier-less gradients		Logarithmic number format	Bilinear interpolation
Dimension of NFA	2	2	1	2
Specified error†	10^{-3}	2^{-10}	10^{-3}	2^{-10}
Error mode	ε_{avg}	ε_{max}	ε_{avg}	ε_{max}
Adder	5	7	5	$\gg 4$[1]
Multiplier	0	0	0	4
Data path width	13	13	11	8
Segments	118	488	132	121

† Error of a single NFA module.

TABLE II
COMPLEXITY RESULTS OF FIVE DIFFERENT NFAS WITH $\varepsilon_{max} = 2^{-10}$ COMPARED TO THE RESULTS IN [9].

	This work		[9]	
Function $f(x_1, x_2)$	No. Segments	Memory (Bits)	No. Segments	Memory (Bits)
$a)$ $\sqrt{x_1^2 + x_2^2}$	488	672	121	9,334
$b)$ $\sqrt[3]{x_1^3 + x_2^3}$	528	564	127	9,658
$c)$ $x_1 x_2 \left(\sqrt{x_1^2 + x_2^2}\right)^{-1}$	512	324	139	10,145
$d)$ $x_1^4 x_2^5$	1066	17,833	193	13,817
$e)$ $\sin(\pi x_1 x_2)$	1064	17,187	263	19,417

for all NFAs. In case of invalid mathematical operations, e.g., division by zero, the corresponding function values are ignored as explained in Sec. III-A.

A. Approximation quality

In a first step, both approximation quality and segmentation effort are analyzed on the algorithmic level. In order to keep the effort reasonable in this paper, only one function (the 2D Euclidean norm function $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ (see Fig. 4)) is considered, varying the QF and the specified (maximum) error. As a symmetric function is regarded, the segmentation reduction technique introduced in Sec. III-B is applied. The input port width is set to twelve bit which causes an internal data path width and output port width of 13 (see Sec. III-B).

As depicted in Fig. 5, QF1 notably achieves the worst performance. All other approaches possess nearly a similar performance, only for very small errors QF2 becomes misaligned. Thus, for the evaluation and comparison of high approximation quality NFAs, QF3 is considered. In all other cases QF2 will sufficiently fulfill the segmentation requirements. As a curiosity, in some cases QF3 requires a lower number of segments than QF4 or QF5. This effect originates from the fact that the reference coefficients β_0' and β_1' are already approximations. Thus, they do not represent necessarily the optimal solution.

B. Algorithmic performance

For the evaluation of the algorithmic performance, two different aspects are considered. First, selected results from the automated function approximation analysis are compared to actual references. This is also done on the algorithmic level. In a second step, five different 2D NFAs are generated and their complexity is evaluated taking the number of segments and the required memory size into account.

In order to achieve meaningful results, our work is compared to two additional NFA approaches which base on the logarithmic number system (LNS) and a bilinear interpolation technique, respectively. As they possess different accuracy requirements, two NFAs using the proposed technique are considered for comparison with both average (QF2, $\varepsilon_{avg} = 10^{-3}$) and maximum (QF3, $\varepsilon_{max} = 2^{-10}$) error constraints. For the average error case, our proposed NFA design achieves slightly better results than the LNS reference in terms of segmentation effort. As the LNS approach requires two NFAs for calculating the Euclidean norm, the average error will be slightly higher. The bilinear interpolation technique achieves very good results in terms of approximation quality. Hence, only a small data path width and a small set of segments is required in order to fulfill the given error constraint. However, these benefits are obtained at the cost of significantly higher arithmetic effort: Due to the comparative complex underlying mathematical method, a large set of additions and multiplications is required[1]. Thus, the algorithmic performance evaluation highlights our work as very efficient in terms of arithmetic calculation effort (see Tab. I).

Next, the complexity is evaluated more generally considering the number of segments and the needed memory space (in Bits) for a larger set of functions (see Tab. II). The results again are compared to the bilinear approach in [9]. For our work, the specified error is set to $\varepsilon_{max} = 2^{-10}$ as well as QF3 is used. Note that the algorithmic effort of the NFAs is not affected by the selected original function. In detail, for the symmetric functions a)-c) and e) again seven additions are required. In order to reduce the total segmentation effort of function e), quarter wave symmetry is exploited and the input operand x_2 is manually adapted (for example, see [10]). As the proposed automated parameter extraction approach considers only point symmetric by now, this is realized by ignoring the function range of $0.5 \leq x_2 < 1$ (see Sec. III-A). The result estimation of the neglected NFA function range can be performed by a simple input operand inversion causing only a minimal additional computational error. As function d) requires no operand swapping, the number of adders can be reduced to a total number of six. The complexity comparison proves our approach as very efficient in terms of memory requirements for nearly all regarded functions, though, the demanded number of segments is higher in all cases. Note that for lower error constraints, which are sufficient in several application areas (for example, see [14]), the segmentation effort decreases evidently (see Fig. 5) and, consequently, even higher performance benefits will be achieved.

[1]Though the total number of adders is not specified explicitly in [9], EVBDD-based segmentation encoder require a high amount of additions in general.

TABLE III
RESULTS OF THE PHYSICAL SYNTHESIS OF THE NFAS COMPARED TO ANALYTICAL IMPLEMENTATIONS.

	Reference	Frequency [MHz]	Complexity [kGE]	Energy [μW/MHz]
a)	**This work**	**660**	**21.02**	**4.20**
	Chipware	406	3.52	8.71
b)	**This work**	**620**	**27.11**	**6.94**
	Chipware		n/a	
c)	**This work**	**620**	**23.71**	**5.36**
	Chipware	221	6.41	21.03
d)	**This work**	**510**	**67.38**	**15.47**
	Generic VHDL	378	13.33	95.56
e)	**This work**	**433**	**76.85**	**20.82**
	Chipware	434	5.85	10.98

C. IC Implementation

In a last step, the proposed 2D NFAs from the algorithmic performance evaluation (see Sec. IV-B) are implemented in CMOS and analyzed. Thus, logical and physical synthesis is performed, considering timing back-annotation as well as parasitic effects. As target technology the 90nm process provided by TSMC is selected. Instead of using memory modules, the required constant values are realized using tie off gates which enable the connection of single nets to fixed logic levels. As the work in [9] does not comprise CMOS-based implementation, our work is compared to analytical implementations that base on either generic VHDL implementations or optimized IP cores provided by the Cadence Chipware design framework [15]. Compared to the analytical implementations, our approach achieves higher performance in terms of throughput[2] and energy consumption in nearly all cases. As for c) no IP core or trivial VHDL implementation is available, no comparison is possible. Considering complexity, our proposal achieves significantly worse results than the analytical approaches. Interestingly, though the results in Tab. II show a dependence on the results from Tab. III, in some cases there are high variations that are not reasonable at first glance, for example the operating frequency of a) and b) compared to b) and c). These effects are explainable by the fact that even small differences for the multiplexer realization (, e.g., the longest sequence of MSBs that must be regarded) directly affect the performance. Hence, these aspects must be taken into account for future work. In order to speed-up the NFAs with a high number of segments, default implementation techniques, for example pipelining [16] must be applied. Further, considering a more complex heuristic at the parameter extraction will lead to higher accuracy or fewer segments most likely. Hence, both aspects must be also regarded in the future.

V. CONCLUSION

In this paper, a novel design method for the efficient generation of two-dimensional (2D) numeric function approximations (NFA) is presented. Significant reduction of the signal processing effort is enabled by the application of both a planar segmentation scheme and function graph approximation by multiplier-less linear equations. As the corresponding hardware architecture is generated automatically, HDL-implementations with only small approximation errors can be realized within short design time. Besides the VHDL-based hardware description, also a testbench and an algorithmic behavioral model can be created which enable hardware verification and simulation, respectively. For evaluation, the results of several different NFAs with varying constraints have been investigated on different levels of the hardware design flow. The proposed approximation scheme, that bases on simplified linear equations, leads to high operation frequencies. The small set of constants lead to small memory requirements, consequently. Summarized, the evaluation highlights our design method for 2D NFAs as a powerful hardware design tool realizing high performance NFAs especially in terms of arithmetic calculation effort, complexity (such as memory requirements), energy consumption and throughput.

[2]As each calculation is performed in one cycle, the throughput (NFA/s) is equivalent to the operating frequency value in Tab. III

REFERENCES

[1] M. Williams, *A History of Computing Technology*. IEEE Computer Society Press, 1997.

[2] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*, 2nd ed. Birkhäuser Bosten, 2006.

[3] P. Brox, J. Castro-Ramirez, M. Martinez-Rodriguez, E. Tena, C. Jimenez, I. Baturone, and A. Acosta, "A Programmable and Configurable ASIC to Generate Piecewise-Affine Functions Defined Over General Partitions," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2013.

[4] Z. Qi, A. Cabe, R. Jones, and M. Stan, "CORDIC Implementation with Parameterizable ASIC/SoC Flow," in *Proceedings of the IEEE SoutheastCon*, 2010.

[5] J. Rust and S. Paul, "Design and Implementation of a Neurocomputing ASIP for Environmental Monitoring in WSN," in *19th IEEE International Conference on Electronics, Circuits and Systems*, 2012.

[6] N. Takagi and S. Kuwahara, "A VLSI Algorithm for Computing the Euclidean Norm of a 3D Vector," *IEEE Transactions on Computers*, 2000.

[7] R. Gutierrez and J. Valls, "Implementation on FPGA of a LUT-Based atan(Y/X) Operator Suitable for Synchronization Algorithms," in *International Conference on Field Programmable Logic and Applications*, 2007.

[8] M. Jansen, R. Baraniuk, and S. Lavu, "Multiscale Approximation of Piecewise Smooth Two-Dimensional Functions Using Normal Triangulated Meshes," *Applied and Computational Harmonic Analysis*, 2005.

[9] S. Nagayama, T. Sasao, and J. Butler, "Programmable Architectures and Design Methods for Two-Variable Numeric Function Generators," *IPSJ Transactions on System LSI Design Methodology*, 2010.

[10] J. Rust and S. Paul, "A Direct Digital Frequency Synthesizer based on Automatic Nonuniform Piecewise Function Generation," in *Proceedings of the 20th European Signal Processing Conference*, 2012.

[11] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2010.

[12] T. Parr, "Enforcing Strict Model-view Separation in Template Engines," in *Proceedings of the 13th International Conference on World Wide Web*, New York, NY, USA, 2004.

[13] J. Rust, F. Ludwig, and S. Paul, "Low Complexity QR-Decomposition Architecture Using the Logarithmic Number System," in *Design, Automation Test in Europe Conference Exhibition*, 2013.

[14] J. Rust, T. Wiegand, and S. Paul, "Design and Implementation of a Low Complexity NCO based CFO Compensation Unit," in *Proceedings of the 20th European Signal Processing Conference*, 2012.

[15] *Chipware IP Components in Encounter RTL Compiler*, Cadence Design Systems, 2010.

[16] H. Kaeslin, *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*. Cambridge University Press, 2008.