On High-Accuracy Direct Digital Frequency Synthesis Using Linear Function Approximation

Jochen Rust[1], Moritz Bärthel and Steffen Paul[1]
[1]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*—This paper describes a novel design method for direct digital frequency synthesizers with very high accuracy. To this end, we leverage the well-known piecewise, multiplier-less function approximation method by two separate enhancements: A parallel function estimation scheme is applied which increases the approximation accuracy and reduces the segmentation effort. To achieve further performance improvement, gradient encoding is also taken into account. For evaluation, several direct digital frequency synthesizer architectures with varying accuracies are generated and analyzed in terms of complexity and timing. Logic and physical synthesis is performed with selected candidates. The results indicate the proposed function approximation technique as a powerful approach for the design of direct digital frequency synthesizers with spurious free dynamic ranges of 90 dBc and more.**

***Index Terms*—Direct Digital Frequency Synthesis, Advanced Linear Function Approximation, Elementary Functions**

I. Introduction and Related Work

In recent years, the design of high-performance Frequency Synthesizers (FS) has become more and more important, as it is used in a vast range of different applications, e.g., medical devices [1] or mobile communications [2]. Its main task is the generation of sine functions with varying frequencies. By now, a large number of different approaches for the efficient hardware-based implementation is available, e.g., Phase-Lock-Loops (PLL), Voltage- (VCO) or Numerically-Controlled-Oscillators (NCO) [3]. A common approach to realize NCOs are Direct Digital Frequency Synthesizers (DDFS), as they have proven to achieve very high performance, especially in terms of power consumption, stability and accuracy [4]. As depicted in Fig 1, a DDFS-related hardware architecture possesses a phase accumulator and a sine mapper. For the realization of the (non-linear) sine function, several design techniques have been explored, targeting an optimal trade-off between hardware performance – in this paper this term comprises timing, complexity and power consumption – and accuracy, assessed by the Spurious Free Dynamic Range (SFDR) [5].

In the last years, high-performance DDFS processing has been realized by the utilization of linear function approximations, mostly extended by well-established improvement techniques like multiplier-less gradients or non-uniform segmentation. For example, in [6] a hand-optimized approach with three fixed segmentation schemes and linear sub-functions is proposed. In [7] the estimation of the (quantized) linear coefficients is enhanced by non-linear mixed-integer linear programming (MILP) optimization methods. Compared to more complex polynomial approaches, e.g., Chebyshev-polynomials [8], very high performance can be achieved, especially in terms of complexity. However, for DDFS architectures with very high accuracy (with an SFDR of 110 dBc and more), e.g., required for GSM-based digital-down-conversion [9], the MILP-based function approximation approach is not practicable, as the computational effort for the parameter estimation is too excessive [7].

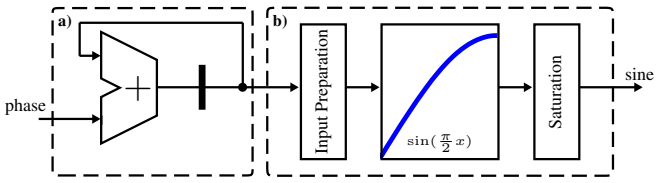

Fig. 1. Schematic overview over a DDFS architecture based on **a)** the accumulator and **b)** the sine mapper. The latter is realized by the approximation of the non-linear quarter sine wave, an operand flipping unit and a result saturate unit catching possible overflows.

To overcome this limitation, an automated function approximation technique has been proposed in [10]. There, a straightforward segmentation is utilized to reach a maximum SFDR of 110 dBc. However, the resulting hardware effort, especially the huge multiplexer, significantly decreases the overall performance which is a major drawback of this approach.

In this paper we will advance this function approximation design technique in order to realize high-accuracy DDFS architectures with reasonable hardware requirements. To this end, we propose the following enhancements:

- Instead of a direct approximation, the original function is split up into a global gradient estimation and a residue that can be processed in parallel (see Sec. II-C).
- To reduce the size of the multiplexer, the gradient selection data is encoded. The decoding, that covers the selection of the true gradients, is processed after the expensive multiplexer tree traversal (see Sec. II-D).

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

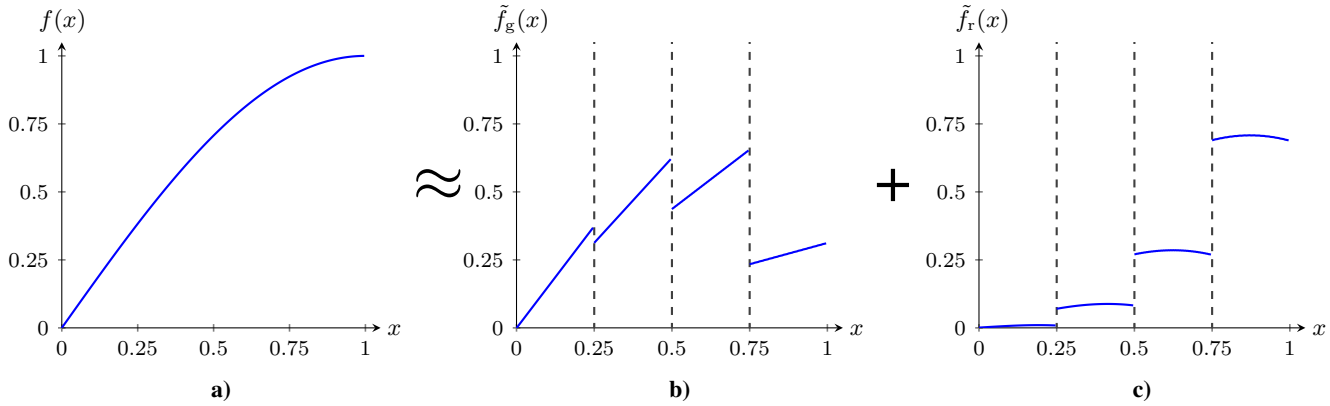

Fig. 2. Example of the proposed function parallelization scheme with **a)** the quarter sine function, **b)** the global approximation $\tilde{f}_g(x)$ for gradient effort reduction and **c)** the residue $\tilde{f}_r(x)$. Note, that **b)** and **c)** can be calculated in parallel and superposed afterwards enabling concurrent signal processing.

II. Advanced Linear Function Approximation

Main idea of the proposed DDFS architecture is the realization of the sine function by means of linear function approximation. In general, it is given by the mathematic formula

$$\tilde{f}(x) = c_1 x + c_0 \ , \tag{1}$$

where x denotes the input data, $\tilde{f}(x)$ the approximation of an original function $f(x)$ and c_0, c_1 the linear coefficients. For advanced linear function approximation, four separate design techniques are proposed which are presented in the following.

A. Multiplier-less Gradients

To decrease the arithmetic signal processing effort, multiplier-less gradients $\tilde{c}_1$ are taken into account. Thus, $\tilde{c}_1$ will contain only small set of accumulated partial products. Its calculation can be realized by trivial shift-and-add means [11]. The total number of partial products is specified by the so-called quantization factor (QF) which refers to the (maximum) number of nonzero digits in the gradient. Out of this, quantized linear gradients can be calculated by

$$\tilde{c}_{1,q} = \sum_{j=0}^{q-1} \pm \left(2^{\lambda_{1,j}}\right) \ , \tag{2}$$

with q as the QF and λ_j as the exponent of the j-th partial product.

B. Non-Uniform Segmentation

For the realization of a non-uniform segmentation scheme, the original function is split up into several sub-functions with variable input ranges. To enable fast access to each sub-function, additional restrictions have to be considered. In detail, a segment must fulfill the constraint

$$\mathrm{seg}(i) = \mathrm{seg}(i-1) + \frac{\mathrm{C}_2 - \mathrm{C}_1}{2^{h_i}} \ , \tag{3}$$

with C_1, C_2 as start and end point of the function, i as the segment index ($\mathrm{seg}(0) = \mathrm{C}_1$) and $h_i \in \mathbb{N}^+$ as the interval exponent of the ith segment. This formula allows the selection of each segment simply by taking the most significant bits (MSBs) of x into account. Note, that h_i may differ for each segment which may cause a varying number of MSBs that has to be considered for each segment. The input range of the original function is set to

$$\mathrm{C}_2 - \mathrm{C}_1 = 2^{h_{\max}} \ , \tag{4}$$

with $h_{\max}$ as the interval exponent of the entire function range.

Along with the multiplier-less gradients from Sec. II-A, the entire function approximation is defined by the following system of equations

$$\tilde{f}(x) = \left[\sum_{j=0}^{q-1} \pm \begin{pmatrix} 2^{\lambda_{0,j}} \\ 2^{\lambda_{1,j}} \\ \vdots \\ 2^{\lambda_{k-1,j}} \end{pmatrix} x + \tilde{\mathbf{c}}_0 \right]^{\top} \cdot \boldsymbol{\kappa}(x) \ ; \ \lambda_{i,j} \in \mathbb{Z} \ , \tag{5}$$

with $\tilde{\mathbf{c}}_0$ containing the offsets of each segment, i and j as segment and partial product indexes, respectively. $\boldsymbol{\kappa}(x)$ is a fade-out function realizing the the segmentation by

$$\boldsymbol{\kappa}(x) = \begin{cases} (1,0,...,0)^{\top}; & \mathrm{C}_1 \leq x < \mathrm{seg}(1) \\ (0,1,...,0)^{\top}; & \mathrm{seg}(1) \leq x < \mathrm{seg}(2) \\ \vdots & \vdots \\ (0,0,...,1)^{\top}; & \mathrm{seg(k-1)} \leq x < \mathrm{C}_2 \end{cases} , \tag{6}$$

with k as the total amount of segments.

C. Parallelization

One of the main drawbacks of piecewise function approximation is the extreme growth of complexity for high-accuracy approximations caused by the huge multiplexer tree for segmentation. To circumvent this limitation, we propose to divide the original function into two different parts: a global approximation $\tilde{f}_g(x)$ and a residue function $\tilde{f}_r(x)$. Roughly speaking, the former minimizes the calculation effort for the resulting function as it realizes a trivial gradient pre-estimation (see Fig. 2b). This task can be interpreted as the

flattening of the original function (see Fig. 2c) that simplifies the approximation effort of $\tilde{f}_r(x)$ and, consequently, leads to a significant reduction of segments. Note, that this technique is only effective for multiplier-less gradients as it only minimizes the number of partial products for gradient calculation in the flattened function.

Though the number of segments for $\tilde{f}_g(x)$ will decrease, the calculation of two separate function approximations also increases the signal processing effort in total. Generally speaking, the complexity of $\tilde{f}_r(x)$ is reduced at the cost of an additional piecewise and multiplier-less (global) function approximation. To keep this overhead reasonable, only an uniform segmentation scheme with a small number of segments will be considered for $\tilde{f}_g(x)$. This divide-and-conquer approach constitutes an advantage for designs with a high number of segments. Hence, a significant increase of the resulting hardware performance can be achieved. A graphical example of this parallelization technique is given in Fig. 2.

D. Encoded Gradient Selection

Besides a reduction of the multiplexer tree size by parallelization, the use of encoded signals for gradient selection will be exploited to achieve a further decrease of complexity and energy consumption. Thus, instead of partial products with the full data path width d, only a reduced selection signal has to traverse through the segmentation multiplexer trees with a maximum data path width of $d_{\text{enc}} = \lceil \log_2(d) \rceil$. For decoding, an extra multiplexer is required that is connected to the reals partial products. An overview of this measure is depicted in Fig 3.

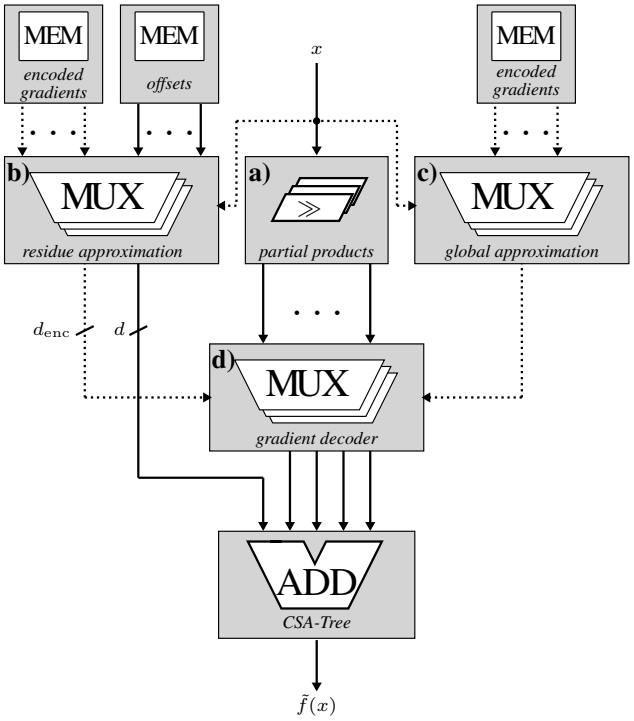

Fig. 3. Hardware architecture of the proposed advanced linear function approximation with **a)** the partial products for multiplier-less gradient calculation (see Sec. II-A), **b)** the multiplexer tree for the non-uniform segmentation of $\tilde{f}_r(x)$ (see Sec. II-B), **c)** the multiplexer tree for the uniform segmentation of $\tilde{f}_g(x)$ (see Sec. II-C) and **d)** the multiplexer for gradient decoding (see Sec. II-D). Solid and dotted lines mark the data and control path, respectively.

III. Hardware Generation

In order to minimize the design time of the advanced function approximation method, an automated, accuracy-driven hardware design method is proposed. In general, it consists of two different tasks, the parameter extraction and hardware mapping.

A. Parameter Extraction

Basically, the parameter extraction is used to calculate the number of segments, its size and location as well as the realization of the linear coefficients. In addition, the advanced approximation techniques proposed in Sec. II are considered. The parameter extraction starts with the parallelization by estimating the global approximation function $\tilde{f}_g(x)$. As multiplier-less and piecewise design techniques must be taken into account at this, a QF (q_{g}) and a number of uniform segments (k_{g}) are mandatory parameters that have to be specified in advance. Within each segment a Best-Case function approximation is performed by the Remez-Algorithm [12] and used as reference. Next, a multiplier-less gradient is calculated by the accumulation of q_{g} partial products that possess a minimal deviation from the reference gradient.

For the calculation of the residue $\tilde{f}_r(x)$, a reference function is set up by calculating the difference between the original function and the global approximation ($\tilde{g}(x) = f(x) - \tilde{f}_g(x)$). As this approximation is allowed to possess non-uniform segments (to enable the demanded accuracy-driven design), a recursive segmentation algorithm is utilized for the approximation refinement. In detail, the segmentation starts with the generation of an approximation considering the whole function range; the gradient estimation of the residue approximation is equal to the one of the global approximation. The offset $\tilde{c}_{r0,i}$ is determined by calculating the smallest sum of absolute errors

$$\tilde{c}_{r0,i} = \underset{\hat{c}_{r0,i}}{\text{argmin}} = ||\tilde{g}_i(x_i) - (\tilde{c}_{r1,i}x_i + \hat{c}_{r0,i})|| \ , \qquad (7)$$

with $\tilde{g}_i(x_i)$ and x_i as the sub-function reference and range of the ith segment, respectively. If the resulting error

$$\varepsilon = \max\left(||\tilde{g}_i(x_i) - \tilde{f}_{r,i}(x_i)||\right) \qquad (8)$$

exceeds the specified error $\varepsilon_{\max}$ the function is divided into two sub-functions by bisection and the segmentation starts over with the leftmost segment. Otherwise, if the condition $\varepsilon \leq \varepsilon_{\max}$ is fulfilled, the parameter extraction continues with the next segment. This straightforward segmentation technique enables a piecewise approximation with varying (non-uniform) segment sizes and finishes, when the entire function range has been processed this way.

As soon as both approximations have been generated, the resulting partial gradients are encoded by assigning each

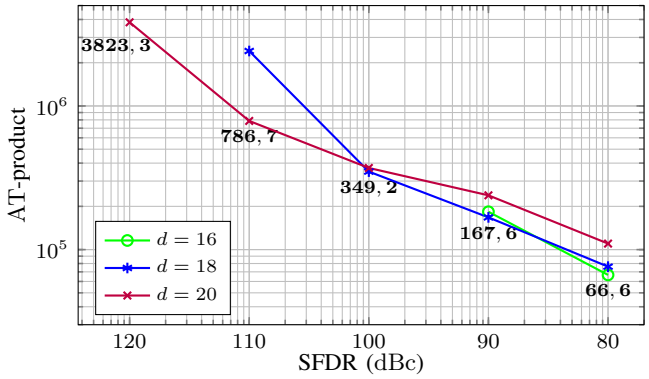

Fig. 4. Design space exploration of the proposed linear function approximation technique by evaluation of the AT-product over the corresponding SFDR for different data path widths. The bold numbers mark the best (Pareto-optimal) candidates.

partial product shift (and the sign) to a related key value. For the decoding, additional multiplexers are necessary, mapping the encoded values back to the partial gradients (see Fig. 3).

B. Hardware Mapping

Besides the parameter extraction, a hardware description of the resulting function approximation has to be set up. For this purpose, VHDL-based template files providing a generic structure of the demanded architecture are used that base on the StringTemplate engine [13]. The input and output ports as well as the constants, encoding values and offset accumulator can be realized by corresponding basic VHDL-expressions. For the quantized gradients, the input operands are shifted accordingly, which is also done by common VHDL means. The segmentation is realized using multiplexers that refer to encapsulated if-statements in VHDL. In order to enable a straightforward StringTemplate processing, the MSB data is transferred to a binary tree data type that is traversed in a depth-first manner during the code generation phase. An architectural example is given in Fig. 3.

IV. Results

To qualify the proposed design technique in the scope of DDFS-based signal processing, an algorithmic and hardware-based evaluation is performed. For this purpose, a quarter sine function is approximated using the proposed advanced function approximation method. By exploiting trivial and well-known design techniques for function mirroring and saturation, the entire sine wave can be processed. Additionally, the range of the sine function is scaled to 1, as this enables a hardware-convenient realization of the function extension, e.g., by bit flipping or signal negation. More details about this measure are given in [10].

TABLE I
Area and time (delay) estimation scheme based on the unit gate model specified in [14] with d and k as the data path width and number of segments (memory cells for MEM or number of adders for ADD), respectively.

	Multiplexer MUX	CSA-Tree ADD	Memory MEM
T (Delay)	$2 \cdot \lceil \log_2(k) \rceil$	$4 \cdot \lceil \log_2(k) \rceil^{\dagger}$	–
A (Area)	$3 \cdot (d \cdot (k-1))$	$7 \cdot d$	$d \cdot k$

† *Only parallel CSA-Tree elements are considered.*

TABLE II
Best-candidate results of the design space exploration, given in Fig. 4, for SFDRs of 90 dBc, 100 dBc and 110 dBc.

SFDR (dBc)	Width d	Residue approximation q_r	Residue approximation k_r	Global approximation q_g	Global approximation k_g
90	18	1	58	2	8
100	18	1	105	2	16
110	20	1	230	2	16

As mentioned in Sec. I, the analysis of a sine function considering its SFDR is a well-established method to value the approximation quality [3]. Unfortunately, estimating the SFDR without the resulting approximation is a very tedious task [7]. Hence, the automated function approximation described in Sec. III is exploited for fast hardware generation enabling exhaustive experimental evaluation of the SFDR.

A. Algorithmic performance

In order to select suitable candidates for further (hardware-based) investigations, a simple but meaningful algorithmic evaluation is performed that base on the well-known Unit-Gate-Model (UGM) [14]. At this, each hardware element of the resulting function approximation architecture is weighted in terms of complexity and timing, according to its internal signal processing effort.

For the timing estimation, the critical path must be taken into account which is determined considering both the global and residue approximation. Hence, it is calculated considering the formulas in Tab. I applied to the architecture given in Fig. 3. Note, that only parallel Carry-Save-Adder (CSA) units are considered for the adder-tree. In order to determine the UGM-based area, again the architectural description from Fig. 3 in conjunction with the values from Tab. I is taken into account.

For exhaustive evaluation, several different specified values for QF: $q_r = \{1, \ldots, 4\}$ (residue approximation), $q_g = \{1, \ldots, 4\}$, number of segments ($s_g = \{1, 4, 8, 16\}$) (both for the global approximation) and the data path width $d = \{16, 18, 20\}$ are taken into account for varying errors $\varepsilon_{\max}$. Note, that $\varepsilon_{\max}$ is refined (starting from $\varepsilon_{\max} = 1$) until the data path resolution is reached ($\varepsilon_{\max} = 2^{-d}$). The SFDR is calculated for each function approximation and evaluated considering the product of timing and complexity (AT-product). The best candidates in the design space are depicted in Fig. 4.

B. IC implementation

For the IC implementation, the best candidates obtained from the algorithmic performance evaluation are taken into

TABLE III
COMPARISON OF IC SYNTHESIS RESULTS TO ACTUAL REFERENCES.

Domain	Reference	SFDR [dBc]	Technology [nm]	QF	Segments	Max. Frequency [MHz]	Norm. Area [10^5]	Energy [μW/MHz]
90 SFDR	**This work**	**90.5**	**130**	$\mathbf{1+2^{\dagger}}$	$\mathbf{58+8^{\dagger}}$	**257**	**9.56**	**5.55**
	[10]	91.7	130	2	141	278	4.38	5.53
	[6]	90.3	130	3	32	216	4.21	5.98
	[7]	88.9	130	3	32	220	3.85	7.42
100 SFDR	**This work**	**101.4**	**130**	$\mathbf{1+2^{\dagger}}$	$\mathbf{105+16^{\dagger}}$	**213**	**12.31**	**8.32**
	[15]	101	250	-	-	201	5.7	61.7
110 SFDR	**This work**	**112.6**	**130**	$\mathbf{1+2^{\dagger}}$	$\mathbf{230+16^{\dagger}}$	**175**	**16.42**	**9.23**
	[10]	110.7	130	3	626	133	17.1	24.89
	[16]	∼110	250	-	-	250	19.2	400

†*The number of both global and residue segmentation is considered by superposition.*

account. To keep the effort reasonable, only three different accuracies are discussed in this paper: $90\,\mathrm{dBc}$, $100\,\mathrm{dBc}$ and $110\,\mathrm{dBc}$. Logical and physical synthesis as well as corresponding verification and timing back-annotation design steps are performed. As target technology a general purpose $130\,\mathrm{nm}$ CMOS process provided by UMC is chosen. In order to enable a fair comparison, the normalized area is used for complexity evaluation that is calculated by dividing the resulting area by the squared technology size [6]. An overview of the synthesis results are given in Tab. III.

The results highlight our approach to be very suitable in terms of operating frequency and energy consumption. Due to the additional signal processing effort, e.g., caused by the additional trial function approximation, only low performance is achieved for an SFDR of $90\,\mathrm{dBc}$ in terms of complexity. However, in higher SFDR domains ($100\,\mathrm{dBc}$ and more), this effect is (nearly) equalized as the multiplexer saving start to dominate the additional hardware effort. Thus, the proposed design techniques result in a very balanced DDFS-architecture, improving the hardware performance in total.

V. CONCLUSION

In this paper, novel architectures for direct digital frequency synthesizers that base on advanced linear function approximation are introduced. Besides multiplier-less linear equations and non-uniform piecewise segmentation − that have already proven to achieve high computational performance for CMOS-based hardware implementations −, also parallelization and encoding techniques are taken into account leading to high-performance and high-accuracy results. The architecture is generated automatically realizing an evident decrease of the design time.

For evaluation, varying accuracies are considered. First, several function approximations are generated and compared in terms of accuracy, timing and area on the algorithmic level of the hardware design flow. In a next step, the IC implementation is performed for selected candidates. As a result, our work achieves very good results for high-accuracy hardware architectures. In conclusion, the design of direct digital frequency synthesizers that base on advanced function approximation have turned out to be a powerful and promising approach. Hence, its application in different application areas has to be considered for future work.

REFERENCES

[1] Y.-X. Wang, W.-M. Chen, and C.-Y. Wu, "A 65nm CMOS low-power MedRadio-band integer-N cascaded phase-locked loop for implantable medical systems," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, Aug 2014, pp. 642–645.

[2] 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, pp. 116–120.

[3] B.-G. Goldberg, *Direct digital frequency synthesis demystified*. LLH Technology Publishing, 1999.

[4] P. Kern, "Direct digital synthesis enables digital PLLs," *RF design*, 2007.

[5] A. Torosyan and A. Willson, "Exact analysis of DDS spurs and SNR due to phase truncation and arbitrary phase-to-amplitude errors," in *Proceedings of the 2005 IEEE International Frequency Control Symposium and Exposition.*, Aug. 2005.

[6] D. De Caro, N. Petra, and A. Strollo, "Direct Digital Frequency Synthesizer Using Nonuniform Piecewise-Linear Approximation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2011.

[7] A. Ashrafi, "Optimization of the Quantized Coefficients for DDFS Utilizing Polynomial Interpolation Methods," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 61, no. 2, pp. 105–109, 2014.

[8] A. Ashrafi, R. Adhami, L. Joiner, and P. Kaveh, "Arbitrary waveform DDFS utilizing Chebyshev polynomials interpolation," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 8, pp. 1468–1475, 2004.

[9] S. Creaney and I. Kostarnov, "Designing Efficient Digital Up and Down," Xilinx Inc., Tech. Rep., 2008. [Online]. Available: http://www.xilinx.com/

[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, pp. 230–234.

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

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

[13] 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.

[14] R. Zimmermann, "Lecture notes on Computer Arithmetic:: Principles, Architectures and VLSI Design," Integrated Systems Laboratory, Swiss Federal Institute of Technology (ETH) Zürich, Tech. Rep., 1999.

[15] D. De Caro, N. Petra, and A. Strollo, "Reducing Lookup-Table Size in Direct Digital Frequency Synthesizers Using Optimized Multipartite Table Method," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 7, pp. 2116–2127, Aug. 2008.

[16] Y. Song and B. Kim, "A 14-B Direct Digital Frequency Synthesizer with Sigma-Delta Noise Shaping," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 5, pp. 847–851, May 2004.