Automated Analog Mixed Signal IP Generator for CMOS Technologies

Mohsen Hassanpourghadi, Qiaochu Zhang, Praveen Sharma, Jaewon Nam, Shiyu Su, Subhajit Chowdhury, Jagannathan Sathyamoorthy, Walter Unglaub, Fangzhou Wang, Mike Chen, Sandeep Gupta, Anthony Levi
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2533
Email: alevi@usc.edu

Wes Hansford
MOSIS
USC Information Sciences Institute
Marina del Rey, CA 90292-6695
Email: hansford@mosis.com

William Taylor
Office of the CTO
Global Foundries Fab8
Malta, NY 12020
Email:
william.taylor@globalfoundries.com

Abstract— **Analog mixed-signal (AMS) modules are typically custom-designed by experienced designers, despite significant effort over several decades devoted to automated synthesis tools. This custom-design process is expensive, has high time-to-market, and often leads to suboptimal design. As part of a DARPA-funded project, we are developing a new approach based on customization of known-good designs (KGDs) to optimally satisfy the designer's intent for an AMS module, expressed in terms of specifications and criteria for optimization. This short paper summarizes some key aspects of the project which provides open-source tools capable of automating analog mixed-signal schematic generation for CMOS technologies.**

Keywords—Open-source automated analog mixed signal circuit design, circuit schematic optimization, automated circuit IP generation

I. Introduction

The automation of analog mixed signal schematic generation has remained a challenge for many years. Our approach is inspired by, and is a generalization of, earlier efforts that used very specific searches to discover new behavior. For example, [1-3], starting with the observation that most analog circuits use a small number of transistors, a numerical study of all possible configurations of a two-transistor circuit was initiated. This research led to the discovery of two low-noise amplifiers, one of which not only provided superior performance but also showed for the first time that thermal noise could be canceled. In our view, a key aspect of this previous successful work is its narrow focus. Our approach also minimizes search by maximizing use of digital synthesis, utilizing knowledge of known-good circuit designs and limiting exploration of parameter space around these point solutions. Our research and development effort is funded by DARPA as part of the POSH program managed by Andreas Olofsson in MTO.

DISTRIBUTION STATEMENT: Approved for public release: distribution is unlimited. This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7853. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL and DARPA or the U.S. Government.

II. AMS Design Methodology

A sketch of the AMS design flow is illustrated in Fig. 1. In step 1 architectures are automatically selected that best match the desired user-performance specification and robustness. The selection process exploits known-good designs (KGDs) from a searchable database, such as phase-locked loop (PLL), digital phase-locked loop (DLL), analog-to-digital converter (ADC), and digital-to-analog converter (DAC) modules and other circuit elements. Step 2 automatically compiles PDK independent module parameters that support the selected circuit architectures. In step 3 a fast circuit generator automatically finds the best combination and parameter settings of the modules without human in the loop. The circuit generator is assisted by an efficient SPICE simulator.

Also shown in Fig. 1 is the user interface that includes access to a repository currently planned to be maintained by MOSIS.

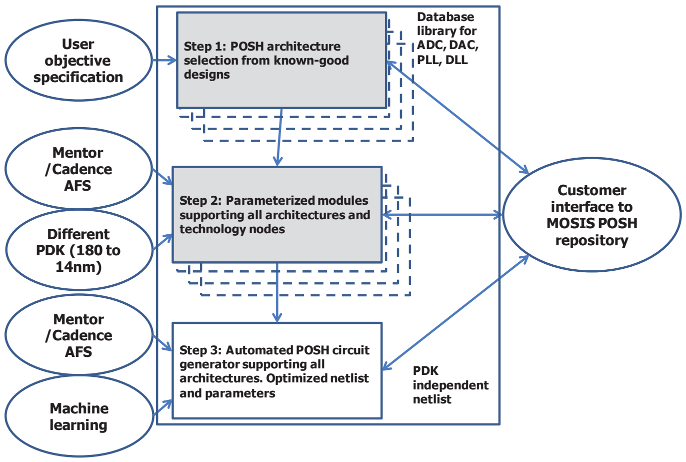

User objective specification
Mentor /Cadence AFS
Different PDK (180 to 14nm)
Mentor /Cadence AFS
Machine learning
Step 1: POSH architecture selection from known-good designs
Database library for ADC, DAC, PLL, DLL
Step 2: Parameterized modules supporting all architectures and technology nodes
Step 3: Automated POSH circuit generator supporting all architectures. Optimized netlist and parameters
PDK independent netlist
Customer interface to MOSIS POSH repository

Fig. 1. AMS design flow showing 3-step process and MOSIS POSH repository resources all of which are planned to be open-source except those requiring foundry NDA.

A. *Exploration near known-good designs*

Starting with record-setting designs by our team [4-10] and other researchers as KGDs, we generalize module structure (viewing it as a set of alternative compositions of sub-modules from a library) and parameters (viewing associated parameter values as algebraic quantities to be customized). We then combine machine learning with search methods to efficiently explore the space of possible customizations of KGDs for a module to obtain module design versions that optimally capture the designer's intent.

Each KGD module is a point in a space defined as a mapping of values from $S \times P$ to values of M, where S denotes the structure of a circuit, $P = \{p_1, p_2, \cdots\}$ denotes the circuit's parameters, and $M = \{m_1, m_2, \cdots\}$ denotes its performance metrics. Our objective is to start with a few KGDs, i.e., specific points in the $S \times P \rightarrow$M space, and to systematically explore a rich set of circuit structures and circuit parameter values to obtain a set of designs that efficiently meet the intent (target specification) of any designer, described by given values of performance metrics and objectives.

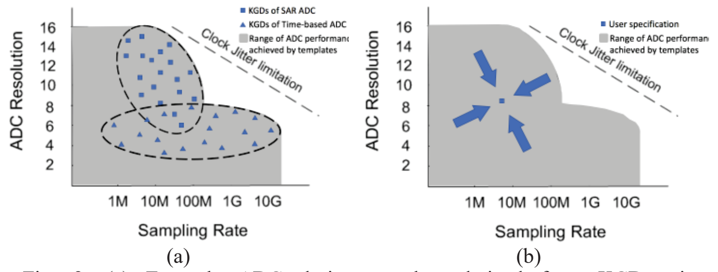


Fig. 2. (a) Example ADC design template derived from KGDs via generalization, to enable (b) rapid creation of a design that meets each design specification, via specialization. This process starts with the architecture templates and user specifications and uses our approach to take rapid decisions to create a suitable ADC design.

Significant experience embodied in KGDs is exploited by incorporating this knowledge into a design template which can be used as a starting point for customization satisfy each design's unique specifications and target fabrication process. As shown in Fig. 2, template creation from a set of KGDs is a generalization of a set of points into a region. In a dual manner, creation of a design that meets a designers specifications from a design template is a specification-directed specialization. Our generalization guarantees that the region covered by the design template for each AMS component covers the range of specifications and fabrication technologies from which a designer may select.

III. AMS Circuit Generator

Fig. 3 shows the block diagram of the proposed automated AMS circuit generator flow. It consists of three major steps. The first step is to prepare the parameterized module library. This stage consists of breaking a KGD into smaller independent modules which serve as building blocks for a parameterized library. We then explore the relationship between design parameters and the performance metrics of the module. For example, the performance metrics of an amplifier module can be gain or bandwidth. Conventional AMS design requires a thorough design exploration through a combination of designer's knowledge and intensive SPICE simulations, which is typically a time consuming process. In our design flow, we explore the possibility of deriving a sufficiently accurate regression model for all the modules, leading to an expedited parameter search process, because the computation requirement for using the regression model is significantly less in comparison to SPICE simulation. To derive the regression model, we first randomly generate a set of training samples via SPICE simulations and use them to train an Artificial Neural Network (ANN) based regression model. The ANN network is implemented in Python using open-source tensorflow, keras and sklearn library, and OCTAVE for post processing.

The second step is to convert user intent into module specifications. AMS specifications demanded by the user are broken down into individual module-level constraints using the system design equations and user's preferred design priorities such as area and power consumption. This step does require designer's knowledge and is coded in Python.

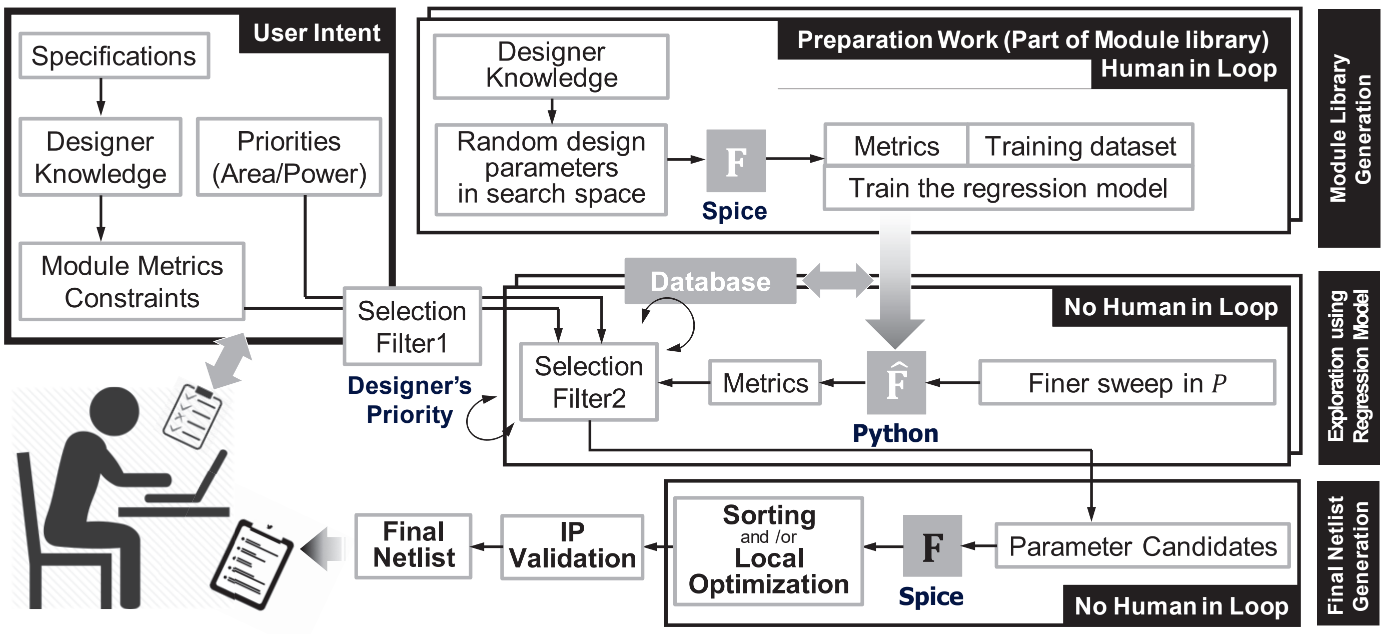
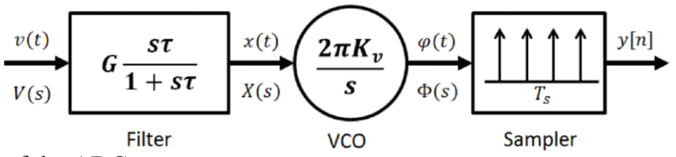


Fig. 3. Machine learning based automated AMS circuit generator flow.

The last step is final netlist generation and parameter optimization. Once the module library is created and user intention is well captured at the module level, the appropriate modules for each block are selected using a selection filter. We use a python script to read the module library and search for modules matching the metrics decomposed in the previous step. Through this process, a few candidates meeting the constraints are selected. They are further validated through SPICE simulation. If the performance is accurate enough and meets the target specification, the best candidate is chosen for generating the final netlist. Otherwise, a local optimization routine can be applied to fine tune the parameter around the selected candidate. Note that, the accuracy of the regression model will determine the local optimization region. This is an effective tradeoff between the complexity of the regression model and local optimization.

IV. Design Example

We use a VCO-Based ADC [11] as an AMS design-flow example. This mostly digital architecture scales well with feature size scaling since it uses time to quantize the input. The general architecture consists of an analog front-end comprising a High Pass Filter (HPF) and VCO followed by mostly digital logic which samples and processes the VCO phase between sampling periods. Fig. 4 shows a high-level functional diagram



of the ADC.

Fig. 4. VCO-Based ADC conceptual diagram.

Having chosen the ADC type, a KGD is chosen as the baseline for further exploration. The block-level architecture of the ADC is shown in Fig. 5. The first step towards automated design is breaking down the architecture into smaller modules that are easier to model and simulate independently and at low computational complexities. The VCO-based ADC is composed of a few relatively simple analog blocks and other mostly digital modules making it a good proof-of-concept candidate.

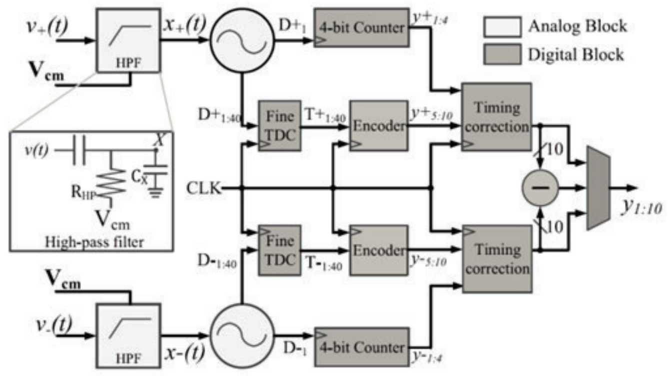


Fig. 5. Architecture of VCO-based ADC from KGD.

The ADC in Fig. 5 has HPF and VCO analog modules. The digital modules are the VCO phase sampler (situated at boundary), Counter, and ROM bubble encoder.

Each module is then parameterized into various design degrees of freedom such as device sizing, bit width, etc., to create a library that can be simulated with varying parameters to explore module performance in the required design space.

The next step is to find the set of design 'parameters' and the set of output behavior 'metrics'. We need to do this for analog modules only since digital circuit synthesis is sophisticated enough to generate circuits using high-level specifications.

The analog blocks are parameterized as in Table. I.

TABLE I. Capturing the Input and Output of the Module Behavioral Model

Module	Design parameters	Output metrics
HPF	Resistance Capacitance	Bandwidth Leakage power Peak attenuation
VCO	Size of inverter Number of stages Bias voltage	Input linear range K_{vco}, $F_{nominal}$, Output amplitude Power dissipation

Since it is not possible to simulate the behavior of the modules across the entire search space, a 'training' set must be chosen that samples the design space, and the remaining space must be estimated using machine learning techniques. Currently the training sample modules are chosen randomly through the search space with ranges derived from KGD heuristics.

TABLE II. Design Parameter Search Range for HPF in the KGD

Parameters	Search range
Resistance	15 Ω ~ 150 Ω (indirectly controlled by poly resistor length and width)
Capacitance	142 fF ~ 4.7 pF (indirectly controlled by varying number of metal fingers (50~288))

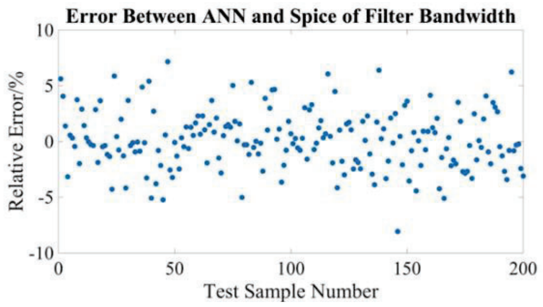
Using a search range indicated in Tables II and III and a random sweep, 1500 training examples are created using the parameterized module library and their performance metrics are captured with SPICE simulation.

TABLE III. Design Parameter Search Range for VCO in the KGD

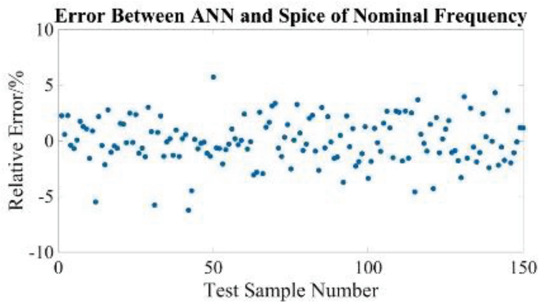
Parameters	Search range
Wp (PMOS size in the unit inverter)	200 nm ~ 600 nm in 100 nm steps
Wn (NMOS size in the unit inverter)	200 nm ~ 600 nm in 100 nm steps
n_fing_p (number of fingers in PMOS)	2 – 20 (random integer number)
n_fing_n (number of fingers in NMOS)	2 – 20 (random integer number)
Zload (load capacitance driven by inverter)	2 fF – 50 fF (random real number)
Vbias (Voltage controlling state on PMOS)	50 mV – 950 mV (random integer number)

The training data set enables an ANN topology to be developed and create an expanded module library which predicts the metric values for modules not present in the training dataset. This allows us to increase coverage of the search space for each module. The neural network has 6-7 input neurons, 20 hidden-layer neurons and 3-4 output neurons.

The neural network is trained on 90% of the data set generated using random samples, and its accuracy is tested using the remaining 10% data samples. Fig. 6a shows the accuracy of the trained model for the filter metric. Fig. 6b shows the accuracy of the model on the VCO metrics. The learning algorithm can also be used for studies into the sensitivity of the individual parameters and can yield important information such as critical and highly sensitive regions which affect the output considerably, or if two parameters are correlated.



(a)



(b)

Fig. 6. (a) Training model accuracy for filter bandwidth using the Neural Network Regressor. (b) Training model accuracy for VCO module parameters.

Once the regression model is derived, the remaining task is to find the constraints of the module metrics using the designer's knowledge described in [11], which converts the top-level specification into module-level requirements. After module specifications are obtained from the high-level design equation breakdown, the library is searched for the candidates for each module of the ADC, and their compatibility is also checked through appropriate boundary conditions.

The design flow is then tested for a few sample designs, and its performance is evaluated using complete system testbenches. Fig. 7 shows the parameter calculation and choices for various high-level specifications.

The final performance is evaluated at both module level as well as entire ADC level to validate the effectiveness of our AMS generator.

1. Module level measurement: The individual module metrics are measured to be within a reasonable range of the expected behavior. In the three test cases simulated, metrics such as filter bandwidth and K_{VCO} are within 3% of the regressor predicted values.
2. ADC level measurement: For the entire ADC, SNR measurement is used to characterize the noise performance. For the three examples tested, the SNR achieved for the 6, 7 and 8-bit ADCs is 42.1, 47.96, and 41.92 dB, respectively.

6 bit 1 GSPS 5715 filter choices 828 VCO choices		7 bit 800 MSPS 8293 filter choices 797 VCO choices		8 bit 700 MSPS 9834 filter choices 481 VCO choices	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
cap_n_h_fing	132	cap_n_h_fing	102	cap_n_h_fing	88
cap_n_v_fing	132	cap_n_v_fing	102	cap_n_v_fing	88
res_width	1u	res_width	1u	res_width	0.5u
res_length	3.5u	res_length	2.8u	res_length	4.2u
N_fing_n	10	N_fing_n	7	N_fing_n	12
N_fing_p	8	N_fing_p	5	N_fing_p	10
Wn	400n	Wn	400n	Wn	200n
Wp	400n	Wp	400n	Wp	200n

Fig. 7. Module parameters for three different top-level specifications.

V. Local Optimization

As indicated in Fig. 3, final netlist generation can include local optimization. If the objective is near a KGD, local optimization may be used to bypass much of the design flow shown in Fig. 3. Hence, this is potentially an alternative way to exploit a growing library of KGDs.

However, local optimization in AMS design can involve multiple objectives. In multi-objective optimization (vector optimization) no single solution necessarily exists that simultaneously optimizes each objective. If the objectives are conflicting, infinitely many Pareto optimal solutions can occur. A Pareto optimal solution exist if none of the objective functions can be optimized without degrading the value of some of the other objective functions. As is well known, the goal then reduces to one of finding a representative class of Pareto optimal solutions and possibly quantifying the trade-offs in satisfying the different objectives [12].

For a given fixed CMOS circuit architecture, there are a finite number of different PMOS and NMOS transistors to optimize, $1 \leqslant i \leqslant N_{\text{PMOS}}$ and $1 \leqslant j \leqslant N_{\text{NMOS}}$ with $n = N_{PMOS} + N_{NMOS}$. Each of the design specifications f_{D} is a function from the parameter space to $\mathbb{C}$, i.e.

$$f_D : \mathbb{R}_{\geq L}^{N_{\text{PMOS}}} \times \mathbb{R}_{\geq L}^{N_{\text{NMOS}}} \rightarrow \mathbb{C} \text{ ,}$$

where L is the minimum parameter for the given process technology file. Given user Inputs = $(y_{D_1}, \ldots, y_{D_k})$, there is a multi-objective optimization problem, namely,

$$\underset{x \in \mathbb{R}^n}{\arg\min}\, \mathbf{F}(\mathbf{x}) = \underset{x \in \mathbb{R}^n}{\arg\min} \left(\boldsymbol{f_{D_1}}, \boldsymbol{f_{D_2}}, \ldots, \boldsymbol{f_{D_k}} \right)$$

where $f_{D_i} = \left(\tilde{f}_{D_i} - y_{D_i} \right)^2$ is a convenient measure.

For gradient method multi-objective optimization a Pareto critical point is approached by $x^{i+1} = x^i + t^i\mathbf{d}^k$, where $t^i > 0$ is the step size and the search direction $\mathbf{d}^i$ is

$$\mathbf{d}^i = \underset{x\in\mathbb{R}^n}{\arg\min}\left\{\max_{j\in\{1,\cdots,k\}} \nabla f_{D_j}(x^i)^{\mathsf{T}}\mathbf{d} + \frac{1}{2}\|\mathbf{d}\|^2\right\}$$

When $k = 1$, the steepest descent direction is $\mathbf{d}^i = -\nabla f_{D_1}(x^i)$ and from Nesterov [13] the gradient is known to decrease at the rate $1/\sqrt{i}$ regardless of starting point, provided the function is convex and Lipschitz continuous. Solving the optimization problem,

$$\mathbf{d}^k = -\sum_{j=1}^{k} \lambda_j^i \nabla f_{D_j}(x^i)$$

where $\lambda_j^i \geq 0$ are the Lagrange multipliers with the constraint $\sum_{i=1}^{k}\lambda_j^i = 1$, giving a direction to carry out multi-objective Nesterov gradient descent [14-15]. A suitable cost function definition is

$$l = \lambda_1 f_{D_1} + \cdots + \lambda_k f_{D_k}$$

which can be shown to have a convergence of ρ_{max}/i, and is the maximum of the Lipschitz constant among all the objective functions f_{D_i}. The solution of the constrained optimization problem is $\left(W_{psol}^i, W_{nsol}^j\right) \in \mathbb{R}_{\geq L}^{N_{\mathrm{PMOS}}} \times \mathbb{R}_{\geq L}^{N_{\mathrm{NMOS}}}$ such that $l\left(W_{psol}^i, W_{nsol}^j\right) < Tol$, where Tol is a user-specified tolerance.

In an initial proof-of-principle demonstration we have shown that local optimization of VCO and PLL circuits can be achieved using the above approach. The current implementation functions best if the number of parameters and the region of parameter search space is limited.

VI. Conclusion

In summary, we have described aspects of our approach to open-source automated analog mixed-signal circuit-schematic generation for CMOS technologies. We have found that a key element to success is maintaining a narrow focus. This includes limiting circuit architectures and maximizing use of digital synthesis. Also, we limit exploration of parameter space to regions that include known-good circuit design. Machine learning and a regression model is found to be an efficient and practical method supplemented, when appropriate, by local multi-objective optimization.

Acknowledgment

The authors wish to acknowledge funding received from DARPA MTO and administered by AFRL contract FA8650-18-2-7853.

References

[1] F. Bruccoleri, E.A.M. Klumperink, and B. Nauta, “Wideband low-noise amplifiers exploiting thermal noise cancellation,” Springer, 2005.

[2] F. Bruccoleri, E.A.M. Klumperink, and B. Nauta, “Noise cancelling in wideband CMOS LNAs,” IEEE International Solid-State Circuits Conference, 2002. Digest of Technical Papers.

[3] F. Bruccoleri, E.A.M. Klumperink, and B. Nauta, “Wide-band CMOS low-noise amplifier exploiting thermal noise canceling,” IEEE Journal of Solid-State Circuits, vol. 39, no. 2, pp. 275-282, 2004.

[4] M. S-W. Chen and R. W. Brodersen, “A 6-bit 600-MS/s 5.3-mW Asynchronous ADC in 0.13um CMOS,” IEEE Journal of Solid State Circuits, vol. 41, no. 12, pp. 2669-2680, 2006.

[5] C-R. Ho and M. S-W. Chen, “A Digital PLL with Feedforward Multi-Tone Spur Cancelation Loop Achieving <-73dBc Fractional Spur and <-110dBc Reference Spur in 65nm CMOS,” IEEE International Solid-State Circuits Conference, pp. 190-192, 2016.

[6] J-W. Nam, M. Hassanpourghadi, A. Zhang and M. S-W. Chen, “A 12-bit 1.6 GS/s Interleaved SAR ADC with Dual Reference Shifting and Interpolation Achieving 17.8 fJ/conv-step in 65nm CMOS,” IEEE Symposium on VLSI Circuits Digest of Technical Papers (2016).

[7] S. Su, , and M. S-W. Chen, “A 12b 2GS/s Dual-Rate Hybrid DAC with Pulsed Timing-Error Pre-Distortion and In-Band Noise Cancellation Achieving >74dBc SFDR up to 1GHz in 65nm CMOS,” IEEE International Solid-State Circuits Conference, pp. 456-457, 2016.

[8] C. R. Ho and M. S-W. Chen, “A Digital Frequency Synthesizer with Dither-Assisted Pulling Mitigation for Simultaneous DCO and Reference Path Coupling,” IEEE International Solid-State Circuits Conference, pp. 254-255, 2018.

[9] C. R. Ho and M. S-W. Chen, “A Fractional-N Digital PLL with Background Dither Noise Cancellation Loop Achieving <-62.5dBc Worst-Case Near-Carrier Fractional Spur in 65nm CMOS,” IEEE International Solid-State Circuits Conference, pp. 394-395, 2018.

[10] S. Su and M. S-W. Chen, “A 16-bit 12GS/s Single/Dual-Rate DAC with Successive Bandpass Delta-Sigma Modulator Achieving <-67dBc IM3 within DC to 6GHz Tunable Passbands,” IEEE International Solid-State Circuits Conference, pp. 362-363, 2018.

[11] P.K. Sharma, and M. S-W. Chen, “A 6b 800MS/s 3.62mW Nyquist AC-coupled VCO Based ADC in 65nm CMOS ,” Custom Integrated Circuits Conference (CICC), Sep. 2013.

[12] S. Boyd and L. Vandenberghe, “Convex Optimization,” Cambridge Univeristy Press, 2004.

[13] Y. Nesterov, “Introductory Lectures on Convex Optimization.” Kluwer Academic Publishers, Dordrecht, 2004.

[14] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. Mathematical Methods of Operations Research, vol. 51, pp. 479–494, 2000.

[15] J. Fliege, A. I. F. Vaz and L. N. Vicente, Complexity of gradient descent for multiobjective optimization, Optimization Methods and Software, 2018, DOI: 10.1080/10556788.2018.1510928.