Fully-Autonomous SoC Synthesis using Customizable Cell-Based Synthesizable Analog Circuits

Ronald Dreslinski, David Wentzloff, Morteza Fayazi, Kyumin Kwon, David Blaauw, Dennis Sylvester
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI, 48109
{rdreslin, wentzlof, fayazi, kmkwon, blaauw, dmcs}@umich.edu

Benton Calhoun
Department of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA, 22904
bcalhoun@virginia.edu

Matteo Coltella, David Urquhart
Arm Holdings plc
Galway, Ireland
{matteo.coltella, david.urquhart}@arm.com

***Abstract*—This paper will show developing a system-on-chip (SoC) synthesis tool which is able to automatically generate a set of analog blocks. Our approach leverages a differentiating technology to automatically synthesize "correct-by-construction" Verilog descriptions for both analog and digital circuits and enable a portable, single pass implementation flow. The SoC synthesis tool realizes analog circuits, including phase locked loops (PLL), power management, analog to digital converters (ADC), and sensor interfaces by recasting them as structures composed largely of digital components while maintaining analog performance. They are then expressed as synthesizable Verilog blocks composed of digital standard cells augmented with a few auxiliary cells generated with an automatic cell generation tool. By expanding the IPXACT format and the Socrates tool from ARM, we then enable composition of vast numbers of digital and analog components into a single correct-by-construction design.**

Keywords— Automatic synthesis tool; SoC; Synthesizable analog block; IPXACT; Differentiating technology

I. INTRODUCTION

Wide usage of system-on-chips (SoC) in different applications such as smartphones, the Internet of Things (IoT), etc. leads to design huge number of such SoCs. As a result, circuit designers spend abundant amount of time on designing SoCs while most of this time is wasting because of being human in the design loop. Integration, debugging, and tuning are just few examples of time consuming tasks in circuit designing which can be performed in notable less amount of time if is implemented by a tool without interference of any human. Furthermore, although it is most likely that another circuit designer has designed exactly or very similar chip that one wants to design, but just because of various numbers of academic papers or datasheets it is impossible for a designer to read and find such similar work and hence remarkable time should be spent on redundant work.

Traditionally one of the most important reasons that analog circuit designing has not been automated yet is impossibility of describing an analog circuit as a human understandable code. As a consequence, in analog circuit design significant amount of designers' time is just being spent on analog chip layout since is not automated as it is on digital part.

Our approach to SoC synthesis is unique in that we leverage a differentiating technology to automatically synthesize Verilog descriptions of analog circuits while maintaining analog performance that are customizable based on user inputs. Here, the analog and digital design blocks are replaced with a single SoC synthesis tool that produces Verilog descriptions for both analog and digital designs. This register-transfer level (RTL) is passed only to the digital chip layout tool for automatic placement and routing (APR) and, therefore no analog chip layout tool is required. On the other hand, with extending the IPXACT format we are able to save and summarize circuit functionality (circuit specs) in extensible markup language (XML) format, which is suitable for searching process and eases finding similar work for circuit designers.

Fig .1 shows a block diagram of the overall program, and how the tasks interact with each other. The SoC synthesis tool takes in user specifications and generates an SoC from models, Commercial Off-The-Shelf (COTS) libraries, and generated analog blocks. When one of the generated analog blocks is required in an SoC, including clock generation, skew correction, data converters, temperature sensors, and memories, the specs for that block will be passed to the analog generation tool that will produce the synthesizable analog design. These analog generation tools function as sub-routines to the SoC synthesis tool, and therefore will be used as a part of the optimization process. Finally, if one of the analog generation functions cannot meet specifications using standard cells, an auxiliary cell is synthesized and added to the cell library. In summary, users will provide their desired functionality and our computer program proposes a finely optimized SoC that meets their specifications and constraints. Such an advantageous tool is highly sought after as it eliminates the need for the cumbersome and tedious process of designing and tuning an SoC.

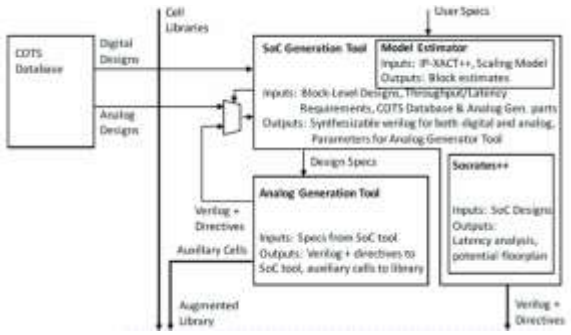

Fig. 1. *Block diagram of the interactions among tasks*

The rest of this paper is organized as follows. In Section II, we describe SoC synthesis tool. In Section III, we present our analog circuit generation methodology. Finally, we conclude in Section IV.

II. SoC SYNTHESIS TOOL

A. Extending IPXACT

Current version of IPXACT just includes information about interface types and parameters, which makes it appropriate for SoC integration tools to have an intelligent configuration, automatic integration, and easy problem debugger. Our goal was extending the IPXACT format to encapsulate all information needed for system generator, and port known parts to the database. This mean IPXACT++, our updated version of IPXACT, was defined for facilitating finding the optimum circuit. For this purpose, this new IPXACT version would include an optional meta-information portion that will contain details about previous implementations of the block (functionality of the circuit) which has not been mentioned in the current version of IPXCAT. This resulted in IPXACT++ having two kinds of information: 1. The information which summarizes circuit functionality, e.g. circuit power consumption. 2. The information which is crucial for appropriate functionality of the circuit, e.g. circuit input voltage range. Saving this information alongside the circuit netlist or Verilog file facilitates the search processing among an extremely large number of circuits to find the most desired one.

In order to ease searching among circuits with the same scope to find the most appropriate one, we make IPXACT++ to contain two parts: general and specific. General would be common in all circuits like area and power consumption values. Specific would be the circuit type in addition to specs values of the circuit, which is different based on the circuit type. Therefore, meta information in the specific part of analog to digital conversion (ADC) circuit differs from phase locked loop (PLL). As an instance, for a circuit that categorized as ADC, "ADC" (as a circuit type) and integral nonlinearity (INL) value are two samples of specific information while for a PLL circuit, "PLL" and jitter value are stored as specific information.

```
<spirit:vendorExtensions>
    <IDEA:circuitFunctionDescriptions>
        <IDEA:circuitFunctionDescription>
            <spirit:name>ADC</spirit:name>
            <IDEA:general/>
            <IDEA:specific>
                <spirit:name>INL</spirit:name>
                <IDEA:values>
                    <IDEA:value>
                        <IDEA:max IDEA:unit="Lsb">
                            <spirit:maximum>1.5</spirit:maximum>
                        </IDEA:max>
                    </IDEA:value>
                    <IDEA:value>
                        <IDEA:min IDEA:unit="Lsb">
                            <spirit:minimum>1.5</spirit:minimum>
                        </IDEA:min>
                    </IDEA:value>
                    <IDEA:value>
```

Fig. 2. *Example of IPXACT++*

B. Automatically scrubbing datasheets for individual pieces of intellectual property (IP) component information

Extracting circuit information requires reading relevant datasheets or academic papers. It is too time consuming to manually do this as the dataset scales to 10M+ parts. So, we need to perform automatic text processing. For this aim we first realize the datasheet title (circuit type class) and then extract relevant data (specs). In both of these approaches, first Portable Document Format (PDF) file is converted to text file which is more suitable for text processing while may causes to lost text connectivity during this conversion.

For the circuit type class realization we use two approaches: machine learning (ML) and Key word searching

There is a well-known model which is called bag of words and is a subset of classification ML approach. In this method, first the text file is segmented into words. Then, occurrence numbers of each word is counted and is assigned as an ID to words. Using a naïve Bayes classifier, with calculated probability coefficients based on training set and using these equations, we can estimate the most probable title of each document using a maximum a posteriori (MAP) approach for testing set.

Since the most important part of supervised ML is having appropriate labeled data, we tried to limit our analyzing region from whole text to a small part of the document that is more likely to mention more important and unique information. This issue become more important when we deal with datasheets as most part of them are tables, numbers, and figures that do not help us because such information do not convey unique information regarding the circuit category. Usually, such unique and important information, which describes the main circuit functionality are written in a section titled such as introduction, circuit description, etc. in both academic papers and datasheets. So, we search for these titles inside the document and if we are successful in finding them on the $i'th$ page, we crop the PDF file to just include $i'th$ and $(i+1)'th$ pages, otherwise we crop the first two pages as it is more likely to cover more important information. Moreover, we divide our training set to two parts: pure-training and test-training sets. There are several options in bag of words and naïve Bayes classifier to use in order to

conclude to the best results. N-gram model which improves inherent orderlessness of bag of words with segmenting text to n subsequent words, Gaussian, Multinomial, and Bernoulli distribution as event model used in Naïve Bayes classifier, and using term frequency–inverse document frequency (TF-IDF) which leads to care more to unique words instead just common, more occurrence words such as "the", "this", etc. and take into account that occurrence number of words is related to the text length (total words) that they are part of it are some examples of these options. So, we train our pure-training set with different combinations of above options and observe results on test-training set using confusion matrix and at the end pick the combination that leads to higher average of percentages in confusion matrix. These approaches improve the accuracy of category realization by 20%.

Alongside of many applications of bag of words model, it has not enough accuracy because of its inherent ML inaccuracy that we want since all of our subsequent tasks are dependent to this circuit category realization. Therefore, we improve this with key word searching method. In the key word searching method, we count the number of each circuit class type name occurrence in each document, pick the maximum, and check whether the relevant specs are mentioned in the document. If the result of these two approaches is not the same, we use an arbitration. Key word searching improves the category realization correctness rate by 15%. Table. I shows confusion matrix of circuit category realization over more than 3000 different PDF datasheets and academic papers.

TABLE I. CIRCUIT CATEGORY REALIZATION CONFUSION MATRIX

Category	*ADC*	*CDC*	*DCDC*	*PLL*	*Temp Sense*	*SRAM*	*LDO*
ADC	97%	2%	0%	0%	0%	1%	0%
CDC	3%	95%	0%	0%	2%	0%	0%
DCDC	0%	0%	98%	0%	0%	0%	2%
PLL	1%	0%	0%	99%	0%	0%	0%
Temp Sense	0%	2%	0%	0%	97%	1%	0%
SRAM	0%	0%	0%	0%	0%	100%	0%
LDO	0%	0%	4%	0%	0%	0%	96%

After determining the document circuit class type, we can start extracting specs from those documents as we should look for specific specs based on circuit category. Information may be reported in either text or tables. We use regular expression to fulfill this goal if the information is as a text. Because of inherent drawback of conversion a PDF to a text file, we first do a text cleaning which includes making all letters to lower case letters to be case insensitive and retrieving paragraphs in the text file as it is in the PDF file. Since there are various ways for reporting different specs, we tried to define our expression such that not be too much tight to include nothing and not be too much general to include everything.

In the table extraction section, it first converts PDF to Comma Separated Values (CSV) file which is more suitable for table processing while there exist lots of imperfection during this conversion. We should find the appropriate row and column that intersection of these two results in target cell. The cell contains "Parameter", "Specification", etc. demonstrates the origin of table which usually locates at top-left corner. Using this cell we can define the territory of each table if there exists more than one table in a page. There exists two mail types of tables as shown in Fig. 3, XY and XX. In XY, desired value, V_i, is located at the intersection of target row, A_i, and target column. Target columns usually have titles such as "this work", "Max", "Min", etc. In XX, "Max", "Min", "Typ" are located at the same row with target values. After recognition table type, we can extract the desired values.

Parameter	Min	Typ	Max
A_1	V_1		
A_2		V_2	
A_3			V_3

(a)

Parameter	Value	Description
A_1	V_1	Min
A_2	V_2	Typ
A_3	V_3	Max

(b)

Fig. 3 *Different table types: a:XY b:XX*

As it was mentioned earlier, there are many defects when a table in PDF file converts to a CSV file. Merging multiple columns/rows, inserting blank column/row at the middle of table, splitting a column/row to multiple columns/rows, etc. are few examples of this imperfection. The most challenging part of table extraction is how to recognize and solve these errors.

As we test our algorithms for data extraction over more than 3000 different PDF datasheets and academic papers, we can successfully extract at least two specs automatically from each document in either text or table extraction.

C. Extending Socrates Infrastructure

The objective of this task is to capture the changes to the IPXACT format into the Socrates SoC design tool. Socrates is a tool that guides a user through the selection and configuration of IP and its assembly into subsystems. Socrates will be extended to be more aware of physical floorplanning and timing and bandwidth constraints. Socrates will also be adapted to provide the necessary details (RTL plus metadata) for the Correct-by-construction wrapper to design systems. The interaction between data recorded in the database, the Socrates configuration file, and the tool itself is important. Initially, the requirements of the design need to be analyzed, including physical data (power, performance and area expectations for the block), technology (process node and variants), floorplan information, and target system definition and configuration options.

D. Correct-by-Construction Design

The objective of this part is to intelligently design systems from high-level user input. The work will be a wrapper around Socrates, that takes as input user block level-designs and approximate throughputs for the system. The tool has two main ways that users can start designing, customized, and recommended. In the customized mode, users will determine exactly the blocks and their specs by their own and connect them together. In this way, the tool is going to find the best candidate for each block. In the recommended mode, users can benefit from existed SoC samples of the tool such as SoC for measuring temperature, pressure or both. Each of these samples contains all blocks, which are needed for appropriate function of the SoC

and their connection. In this mode, user will specify the sensor specs and high level specs for whole SoC. In other words, even though users can determine each block specs separately they can, at the same time, just specify the overall SoC specs such as total power or area.

In general, our COTS database is divided into four regions: fixed components e.g. board regulators that needs to be mined from existing sources as described in the section B, analog generators e.g. parameterized ADC, instances i.e. an instance circuit that is output of analog generator that we used once before, and database of synthesis and placement and routing (P&R) directives i.e. a circuit that has been generated by the whole system once. Fig. 4 shows this COTS database and their interactions.

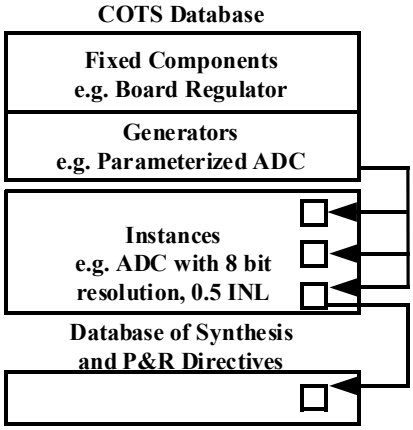

Fig. 4 *COTS database*

The searching process begins with fixed components. As shown in Fig. 2, using max and min we can cover intervals during our searching process. Therefore, even with tons of circuits, it takes short time to categorize them regarding to their circuit functionality and inside each category they can be sorted from smallest to largest respectively the first to the last spec value which dramatically eases the searching process. If we are not able to find the desired circuit in the target technology node, the tool will call analog generator if the circuit category is included in its domain. Otherwise, it tries to estimate the throughputs using technology model estimator. In both conditions, it will populate the database with the generated circuit or estimated values in IPXACT format since calling, generating or estimating are time consuming tasks in comparison to just searching among existed files. Once everything including whole SoC schematic, and P&R is done we will add this SoC to our COTS database to use it in the future if it is needed and not repeating this entire process again.

The inputs will then be used to generate several candidate architectures that will be fed through Socrates for analysis. Fig. 5 shows how these candidates are found. The user will be asked to select optimization priorities from the constraints. This maybe area or power. The tool first tries to find the blocks with minimum area/power. If the summation of power/area for all blocks is larger than our budget which defined as a constraint by the user, it means this SoC with such constraints is infeasible. Otherwise, it checks the other constraints. If the suggested SoC satisfied them it means the tool is done. If not the tool needs to do compromising i.e. degrading the first priority spec of each block in order to improve the other constraints. A final design RTL will be generated along with associated timing constraints and relative/structured placement scripts to be used in synthesis/APR. The design will also identify potential timing critical paths and insert latency insensitive interfaces that allow register slices to be added if timing cannot be met. This information can be fed to automated synthesis and APR scripts to insert register slices when appropriate. In addition, the wrapper will also output the configuration parameters for the analog blocks in the system. The analog tools will be run and candidate designs fed back to the wrapper. The final analog design will be output as a structured netlist for the block.

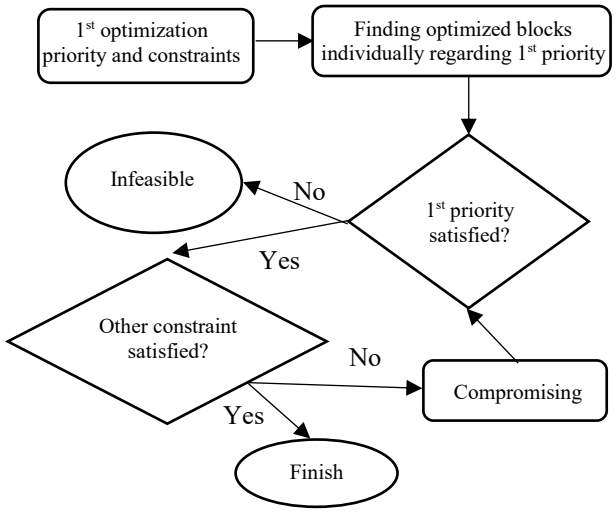

Fig. 5 *Constraint satisfaction algorithm*

III. ANALOG GENERATION

Analog circuit design using authentic analog architecture requires delicate layout and sizing of transistors, which are yet ready to be realized with automation flow. The cell-based architectures, on the other hand, are promising for portability and scalability in terms of analog performance and layout generation since the circuit aspects are controlled by the number of cells being used and are capable to utilize existing digital APR tools. This makes cell-based analog block a credible candidate for automated analog circuit generation, which we use in our flow. The cell-based analog blocks are proven in its performance by various groups [1-6]. Synthesizable all digital phase locked loop (ADPLL), for particular, have been made by full digital flow with APR tool [1][6]. Since these blocks are described in Verilog description and layout constraints, the objective of our Analog Generation tool is to provide RTLs and APR descriptions of each block that support user given specifications. The design flow is identical for all supported analog blocks, and is divided in to two steps: 1. Modeling, 2. Deliverables

Generation and Verification. The flow is depicted on Fig 6. Modeling is a procedure of capturing characteristics of specific Aux-cell and process design kit (PDK), building a relationship between design and specifications. This procedure takes comparably long time due to numerous simulations and is done once per PDK. Deliverables generation & verification stage is repetitive for new user specifications. This will be conducted repeatedly for new specifications. New designs are stored in COTs library, and if an existing design supports a given spec, it will be immediately pulled out, reducing the time of the task. The details of each step are discussed below.

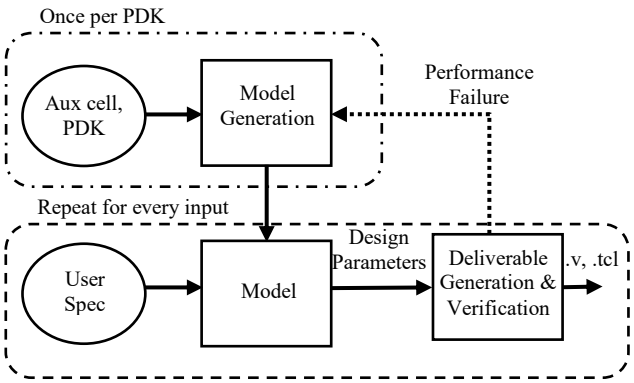

Fig. 6 *Block diagram of analog generation*

A. Model Generation

This part of the task is to capture the analog characteristics of aux-cell of certain PDK by gathering sufficient simulation results, and utilize it to predict optimum design for a given spec. For this, the tool automatically generates test environments, mainly netlists and testbenchs, run the simulation, read the results and use it to build a model. There are two methodologies to make a model from simulation data: 1. Mathematical equation 2. Machine Learning. Mathematical equation based model is to use prior knowledge about circuit to link design and spec. For digitally controlled oscillator (DCO), used in ADPLL, for example, the average ratio of current and capacitance (I/C) of each stage determines the delay of one stage and the frequency is determined by this delay and the number of stages. With this knowledge, we can obtain the average ratio (I/C) by back-calculating the frequency results. This equation-based model holds valid as long as the circuit operates within the property we defined, shows solid accuracy for the frequency of DCO (error < 0.8%, 850 results). Thus, the model is reliable on predicting the frequency of certain design. But to find the optimum design for a set of specs(frequency range, frequency resolution, phase noise, jitter, area etc.) using the mathematical model, the tool designer has to build an algorithm that narrows down the design space and pick one that satisfies all the spec constraints. This is limited since the algorithm relies on the equations and designer intelligence. The algorithm relates the equations for each spec and sequentially narrows down design parameters, using the inequalities to satisfy the specs. This human-built algorithm requires update for additional spec or design parameters. Thus, equation-based algorithm is limited in terms of both reliability and scalability.

Machine learning can be used to find the design solution for a given spec, or the other way(spec of certain design), by training it with sets(design, spec) of data. Once large sample data is used for training, it can predict a design that is likely to have a specific spec. If the design was in the range of that was used for training, machine learning shows high accuracy. Table 2 shows the accuracy comparison of equation based and ML based for predicting 4 DCO specs of 55 different designs. There are specs that follows the equation accurately, while some shows weak accuracy. ML shows reliable accuracy over all the specs. It is robust in terms of the number of inputs (specs) and outputs (design parameter), since it only needs new sample data to be trained, unlike equation-based algorithm has to be renewed relying on designer intelligence. It is also strong in predicting a solution of specs with high equation complexity, while equation-based algorithm has limits due to the error of the equation itself. The difficulty of utilizing ML is to generate massive simulation results for the learning data, and to define the valid range of prediction.

To summarize the equation-based model, relatively fewer simulation results are required since it has strength on extrapolation. However, the equation-based design searching algorithm has limitations because it relies on complex equations and designer intelligence to link those. The ML based model, on the other hand, requires huge simulation results to cover wide range with solid accuracy, but once the reliable model exists, it is easy to find a design solution accurately, and number of inputs and outputs are scalable. Thus to leverage the strength of both methods while covering each other's weakness, we will use the equations to generate sample data for ML without simulation and use the trained ML to predict the design solution. This way, the tool requires few simulations to build a model, and has a strong design solution prediction.

TABLE II. MAXIMUM ERROR RATE COMPARISON OF DCO SPECS BETWEEN MACHINE LEARNING AND EQUATION BASED MODEL

Max Error (%)	f_{max}	f_{min}	Δf_{res}	Area
Machine Learning	1.11	2.39	1.66	1.37
Equation model	0.35	0.64	8.83	0

B. Deliverable generation and Verification

This step is to convert the design parameters decided in the modeling stage to a Verilog format and verify the performance with the layout parasitic included. To automate the procedure, the design parameters are written in parameterized form in RTL descriptions so that the tool can simply replace those parameters with the set the modeling stage provided. The synthesis and APR scripts are written in the same way to ease automation.

The key challenge of using APR for cell-based analog block layout is the induced systematic mismatch due to the random placement and interconnects. There has been prior research on calibrating and utilizing the mismatch to enhance control resolution [6]. Another way to deal with the issue is to constrain the geometry of cell placement in the way that minimizes the mismatch. This method has limited portability for different aux-cells since the geometry should change according to the size of aux-cell. We are in the stage of deciding the method to address

the issue, comparing strengths and weaknesses of various techniques. Until now, we have used different power domains to allocate limited area for placement of analog blocks. This can reduce the parasitic effects by reducing interconnect metal length.

Once the layout is done, the tool automatically runs a post extraction (PEX) simulation, which includes the layout parasitic effects, to verify the performance. If the performance meets the user specification, the tool will provide the RTL, APR descriptions to the next stage. If it fails to meet the specs, it will iterate the design process by tweaking the design variables that improve the bottleneck specs. It will also modify the model, to increase the margins of certain spec for layout parasitic.

IV. CONCLUSION

In this paper, we showed how to develop an autonomous SoC synthesis tool that intelligently generate an SoC based on user input. Using our COTS database the tool can find the most optimized circuit which fulfill user's requirement. One of the unique features of this proposed program are that both analog and digital designs are described in Verilog and are able to be automatically placed and routed. Therefore, any synthesis tool created as a part of Verilog, cell libraries, and directives to guide the groupings of cells can be used to complete the physical design of our fully synthesized SoC.

ACKNOWLEDGMENT

This material is based on research sponsored by Air Force Research Lab (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7844.

REFERENCES

[1] W. Deng, D. Yang, and A. Matsuzawa, "A Fully Synthesizable All-Digital PLL With Interpolative Phase Coupled Oscillator, Current-Output DAC, and Fine-Resolution Digital Varactor Using Gated Edge Injection Technique," *IEEE JSSC*, vol. 50, pp. 68-80, Jan 2015

[2] S. Bang, D. Blaauw, and D. Sylvester, " A Fully Integrated Successive-Approximation Switched-Capacitor DC-DC Converter with 31mV Output Voltage Resolution," *IEEE ISSCC Dig. Tech. Papers*, pp. 370-372, Feb 2013

[3] K. Yang, D. Blaauw, and D. Sylvester, " A 0.6nJ -0.22/+0.19°C Inaccuracy Temperature Sensor Using Exponential Subthreshold Oscillation Dependence," *IEEE ISSCC Dig. Tech. Papers*, pp. 160-161, Feb 2017

[4] M. Shim, D. Blaauw, and W. Jung, " An Oscillator Collapse-Based Comparator with Application in a 74.1dB SNDR 20KS/s 15b SAR ADC," *IEEE VLSI*, Jun 2016

[5] W. Jung, D. Blaauw, and D. Sylvester, "A 0.7pF-to-10nF Fully Digital Capacitance-to-Digital Converter Using Iterative Delay-Chain Discharge," *IEEE ISSCC Dig. Tech. Papers*, pp. 484-486, Feb 2015

[6] Y. Park and D. Wentzloff, "An All-Digital PLL Synthesized from a Digital Standard Cell Library in 65nm CMOS," *IEEE CICC*, Sep 2011