Fully-Autonomous SoC Synthesis using Customizable Cell-Based Analog and Mixed-signal Circuits Generation

Tutu Ajayi[1], Sumanth Kamineni[2], Morteza Fayazi[1], Yaswanth K Cherivirala[1], Kyumin Kwon[1], Shourya Gupta[2], Wenbo Duan[1], Jeongsup Lee[1], Chien-Hen Chen[2], Mehdi Saligane[1], Dennis Sylvester[1], David Blaauw[1], Ronald Dreslinski Jr[1], Benton Calhoun[2], and David D. Wentzloff[1]

[1] University of Michigan, Ann Arbor, MI
[2] University of Virginia, Charlottesville, VA

Abstract. This chapter presents the world's first autonomous mixed-signal SoC framework, driven entirely by user constraints, along with a suite of automated generators for analog blocks. The process-agnostic framework takes high-level user intent as inputs to generate optimized and fully verified analog and mixed-signal blocks using a cell-based design methodology.
The approach is highly scalable and silicon-proven by an SoC prototype which includes 2 PLLs, 3 LDOs, 1 SRAM, and 2 temperature sensors fully integrated with a processor in a 65nm CMOS process. The physical design of all blocks, including analog, is achieved using optimized synthesis and APR flows in commercially available tools. The framework is portable across different planar and FinFET CMOS processes and requires no-human-in-the-loop, dramatically accelerating design time.

Keywords: analog synthesis, analog generator, SoC generator

1 Introduction

There is an ever-growing need for automation in analog circuit design, validation, and integration to meet modern-day SoC requirements. Time-to-market constraints have become tighter, design complexity has increased and more functional blocks (in number and variety) are being integrated into SoCs. These challenges often translate to increased manual engineering efforts and non-recurring engineering (NRE) costs. FASoC is an open-source[3] framework for Fully-Autonomous SoC design [1, 2]. Coupled with a suite of analog generators, FASoC can generate complete mixed-signal system-on-chip (SoC) designs from the high-level user specifications. The framework leverages differentiating techniques to automatically synthesize correct-by-construction RTL descriptions for both analog

[3] Source code for the framework and all generators developed as part of this work can be downloaded from https://github.com/idea-fasoc/fasoc

and digital circuits, enabling a technology-agnostic, no-human-in-the-loop implementation flow.

Analog blocks like PLLs, LDOs, ADCs, DC-DC converters, and sensor interfaces are recasted as structures composed largely of digital components while maintaining analog performance. They are then expressed as synthesizable Verilog blocks composed of digital standard cells and auxiliary cells (aux-cells). The framework employs novel techniques to automatically characterize aux-cells and develop models required for generating bespoke analog blocks. The framework is portable across processes and scalable in terms of analog performance, layout, and other figures of merit.

The SoC generation tool translates user intent to low-level specifications required by the analog generators. The IP-XACT [3] standard is leveraged to achieve full SoC integration. Added vendor extensions capture additional metadata relating to the generated blocks. This enables the composition of vast numbers of digital and analog components into a single correct-by-construction design. The fully composed SoC design is finally realized by running the Verilog through synthesis and automatic place-and-route (APR) tools to realize full design automation.

2 Overview

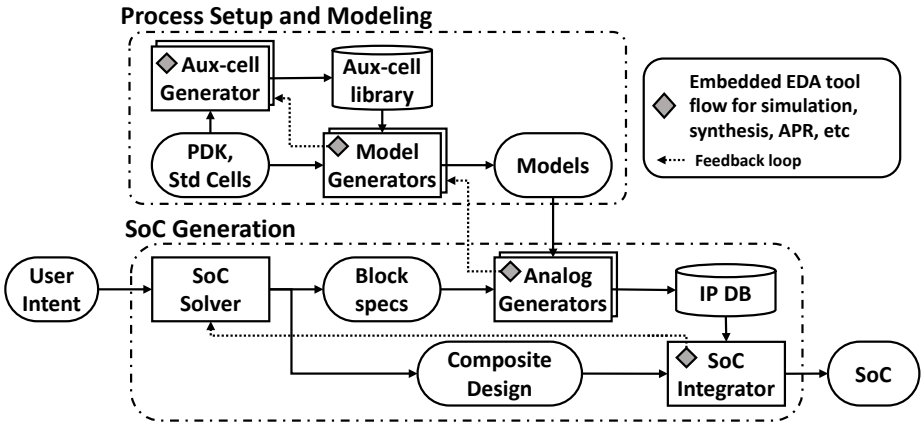

Fig. 1. FASoC Framework Overview [1]

A high-level representation of the framework is shown in Fig. 1. The *Process setup and modeling* phase is performed once for the process design kit (PDK), and it involves the generation of the aux-cells and models for the generator. The setup process is largely dominated by simulations using the design templates in combination with the characterization scripts. The automation of aux-cell and model generation significantly reduces porting effort across PDKs.

The *SoC generation* phase begins by invoking the *SoC solver* repetitively to translate the high-level user-intent into analog specifications that satisfy the user constraints. The *SoC solver* explores the SoC design space through a combination of mathematical and heuristic system models, to determine the necessary system blocks and their specifications. For instance, it can determine the operating frequency of the PLL to be generated based on the target application, SoC. If targeting a neural network application, it considers parameters such as the supported pipelines, instructions per operation, inference per second and operations per inference. The block generators are invoked as needed and the SoC integrator stitches the composed design and walks it through a synthesis and APR flow to create the final SoC layout. The FASoC framework is tightly integrated with analog generators for PLL, LDO, temperature sensor, SAR ADC, switched-capacitor DC-DC converter and SRAM blocks. Section 4 describes the circuit architecture adopted by the different generators.

3 Process Setup and Modeling

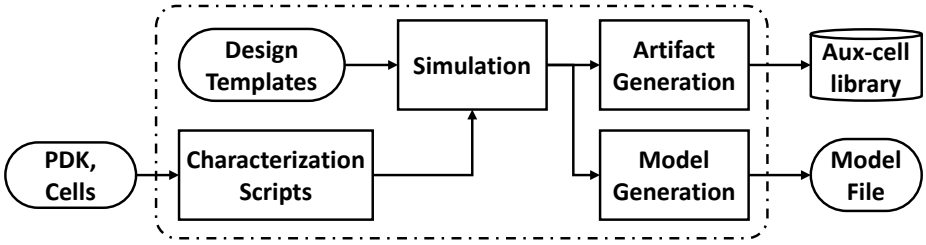

Fig. 2. Aux-cell and model file generation flow [1]

FASoC employs a synthesizable cell-based approach for generating analog blocks, significantly cutting back on manual layout and verification efforts. Aux-cells are small analog circuits that buttress the standard cell library and provide specific analog functionality required by the generators. Each cell is no larger than a D flip-flop and can be placed on the standard cell rows. The creation of aux-cells is simplified by using a suite of design templates in tandem with PDK characterization scripts. The templates capture the aux-cell's precise circuit behavior without including any PDK-specific information. The characterization scripts operate on the PDK to derive technology-specific parameters required to set knobs within the templates. Example parameters extracted from the PDK include threshold voltage, metal parasitics, MOSFET behavior, and Fan-out of 4. The knobs set within the template include device type, transistor sizing, and other circuit design options. The results from aux-cell generation include the netlist, layout, timing library, and other files required to proceed with conventional synthesis and APR. Presently, the layouts for the aux-cells are manually

created, however, there is an expanding array of tools [4–6] for layout automation that show promising results. The template-based methodology for creating aux-cells enhances process-portability and significantly cuts down on design time. All of the generators presented in this work leverage a suite of aux-cells that are depicted in Fig. 3.The template-based methodology enables users to extend the aux-cell library by creating a design template that captures the respective aux-cell precise behavior without including the PDK information.

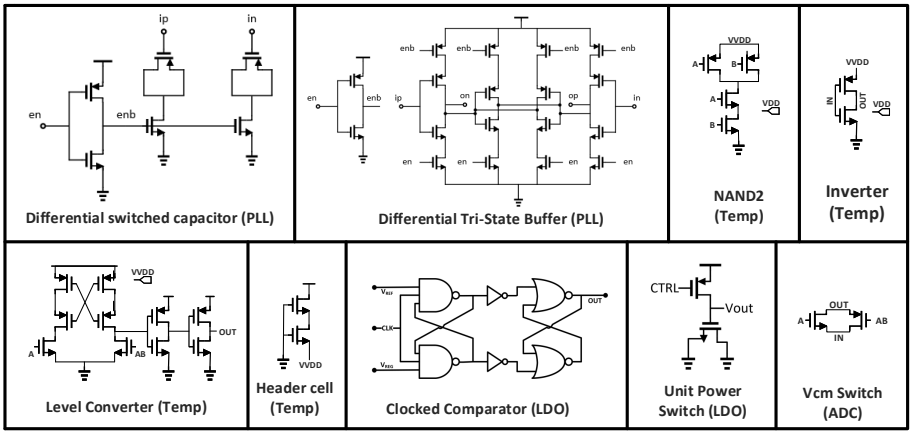

Fig. 3. Schematic for aux-cells used across PLL, LDO and temperature sensor generators [1]

The analog generators use models to predict performance and select design parameters to create optimized block designs that satisfy the input specifications. The models are derived from the parameterized templates that incorporate the aux-cells. The models for each generator vary and are developed from a combination of mathematical equations, machine learning, and design space exploration. The modeling exercise is also performed once per PDK and the results are saved into a model file. Sections 4 briefly describes the modeling approach adopted by each generator integrated into the framework.

4 Analog Generator Architecture

Synthesizable analog blocks [7] were introduced a few decades ago and have continued to evolve, closely matching the performance obtainable by full custom designs. Prior works have described techniques for synthesizing analog blocks for UWB transmitters [8], PLLs [9], DACs [10], and other types of analog blocks [11–13]. This approach lowers engineering design costs, increases robustness, eases portability across PDKs, and continues to show promise even at advanced process nodes [14–16]. The analog generators developed as part of this

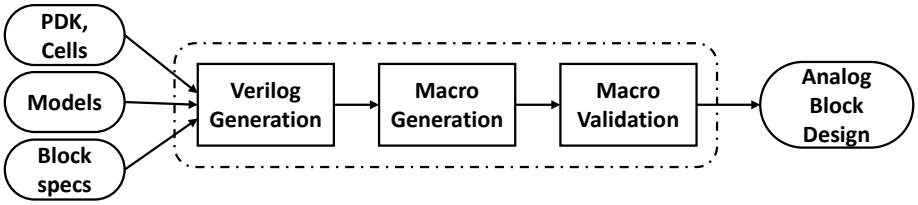

Fig. 4. Analog generator flow [1]

work can be likened to ASIC memory compilers that take in a specification file and produce results in industry-standard file formats, which can then be used in standard synthesis and APR tools. Unlike typical memory compilers, the generators are open-source, process agnostic, and share a scalable framework amenable to different types of blocks. The framework is modular and share a similar process as depicted in Fig. 4. The full generation process is broken down into three steps:

Verilog Generation: This step leverages models to produce a synthesizable Verilog description of the block that conforms to the input specifications. It also generates guidance information in a vendor-agnostic format. The guidance includes synthesis constraints, placement instructions, and other information that may be required by the synthesis and/or APR tool to generate blocks that achieve the desired performance. In addition, this step also reports early estimates on performance and the characteristics of the block to be created.

Macro Generation: The Verilog and guidance information is passed to a digital flow to create macros that can be embedded into larger SoC designs. The digital flow in this step performs synthesis, APR, DRC, and LVS verification. The digital flow includes an adapter to translate the guidance into vendor-specific commands used in synthesis and APR. The adapter abstraction allows us to (1) express additional design intent without exposing protected vendor-specific commands and (2) easily support multiple EDA tools including open-source alternatives [17–19]

Macro Validation: The last step is a comprehensive verification and reporting of the generated block. The full circuit goes through parasitic extraction, SPICE simulations, requirement checks, and other verification to culminate in a detailed datasheet report.

The generators can be invoked standalone, outside of the full SoC generator flow. To simplify the system integration, the AMBA™ APB protocol was adopted as the register interface to all blocks.

The following subsections briefly describe the analog generators currently integrated into the FASoC framework.

4.1 PLL

The generated PLLs (Fig. 5) share the same base architecture as ADPLL [20]. The phase difference of the reference and output clocks are captured by the

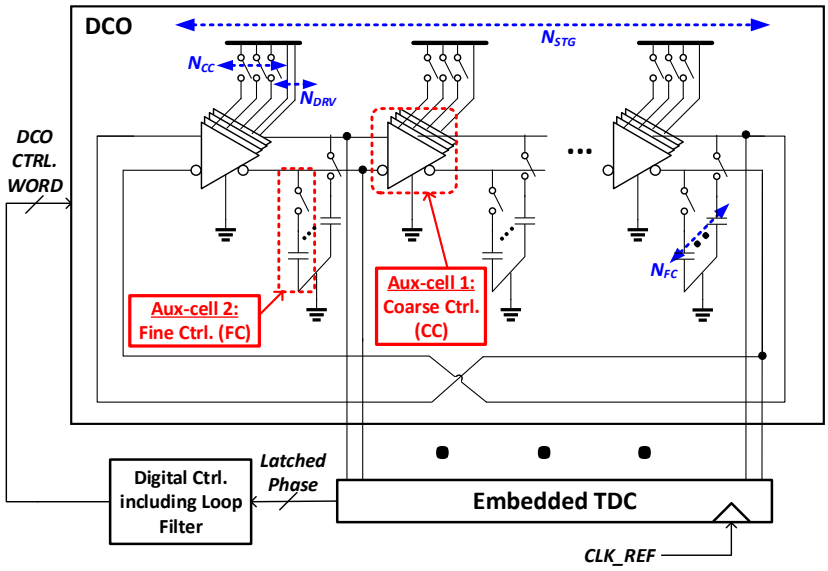

Fig. 5. DCO architecture indicating the aux-cells and designs parameters [1]

embedded time-to-digital converter (TDC), while the digital filter calculates the frequency control word for the digitally controlled oscillator (DCO). The input specification to the generator defines the nominal frequency range and in-band phase noise (PN). The PLL generator uses a physics-based mathematical model [21] for characterization. The first step is building a mathematical relationship between DCO design parameters (number of aux-cells and stages) and the required DCO specifications. Using simulation results from a parametric sweep, the effective ratio of drive strength and capacitance can be derived for each aux-cell. This ratio enables us to predict frequency and power results (frequency range, frequency resolution, frequency gain factor, and power consumption) given a set of input design parameters.

4.2 LDO

The generated LDOs (Fig. 6) share the same base architecture as DLDO [22]. The LDO leverages an array of small power transistors that operate as switches for power management. Based on design requirements, the generator can swap the clocked comparator with a synthesizable stochastic flash ADC [23] to improve transient response. The input specifications to the LDO generator are the V_{IN} range, $I_{load,max}$ range, and the dropout voltage. The generator uses a poly-fit model of the load current ($I_{load,max}$) performance for various combinations of aux-cell connections (connected in parallel and for different VDD inputs) in both ON and OFF states. The model is created by simulating various test circuits after parasitic extraction.

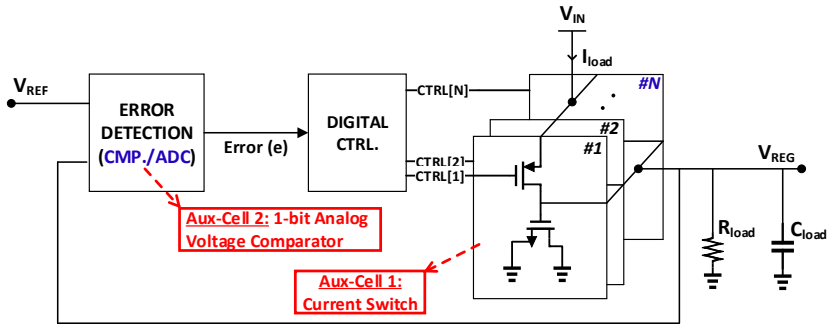

Fig. 6. LDO architecture indicating the aux-cells and design parameters derived from input specifications of V_{IN} , I_{load} and desired transients [1]

4.3 Temperature Sensor

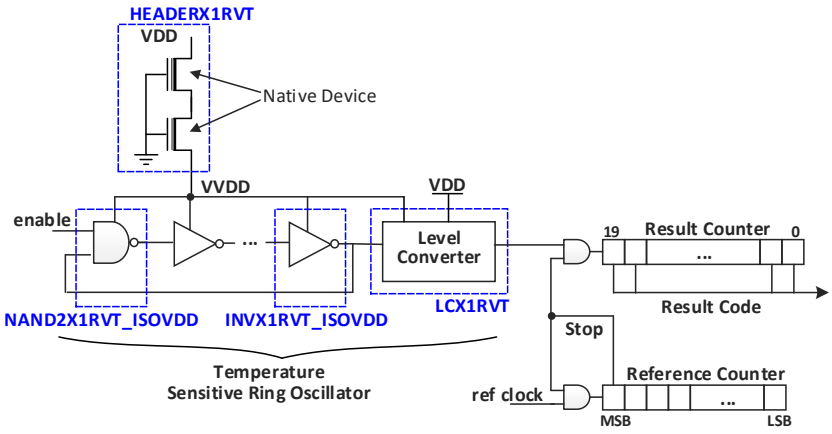

Fig. 7. Temperature sensor architecture indicating the aux-cells [1]

The generated sensors (Fig. 7) share the same base architecture as [24]. The sensor relies on a temperature-sensitive ring oscillator and stacked zero-VT devices for better line sensitivity. The input specifications include the temperature range and optimization strategy, for either error or power. For a given temperature range, the generator first checks a modeling file to select an optimized design model. If the modeling file is not already present, the generator will sweep the design parameters and start the internal simulations. The results are then utilized to train a predictive Bayesian neural network model. Doing this can considerably reduce the total simulation time and predict the best design parameters that can match the input specifications.

4.4 SAR ADC

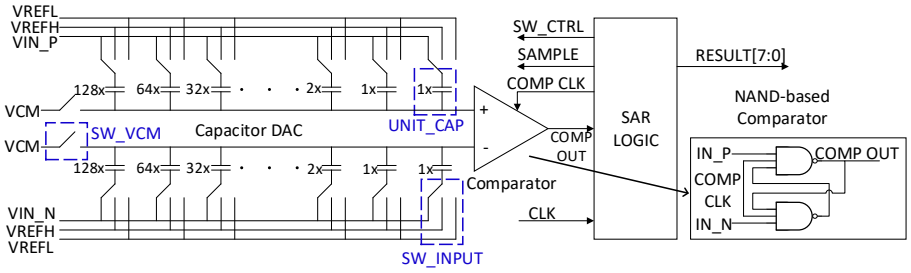

Fig. 8. SAR ADC architecture indicating the aux-cells

The generated SAR ADC (Fig. 8) utilizes the same base architecture as [25], which consists of capacitors, switches, a comparator, and a SAR controller. The Capacitor DAC includes the capacitor arrays and switches, it samples the differential signals at the input. The comparator uses a NAND-based structure, which is more suitable for design synthesis. The SAR controller generates the control signals for switches in the Capacitor DAC and the comparing clock for the NAND-based comparator. The input specification includes the sampling frequency, the target effective number of bits (ENOB), and the optimization method (for either power or area). For a given ENOB value, the generator selects the optimal number of switches that can satisfy the target sampling frequency.

4.5 Switched-capacitor DC-DC converter

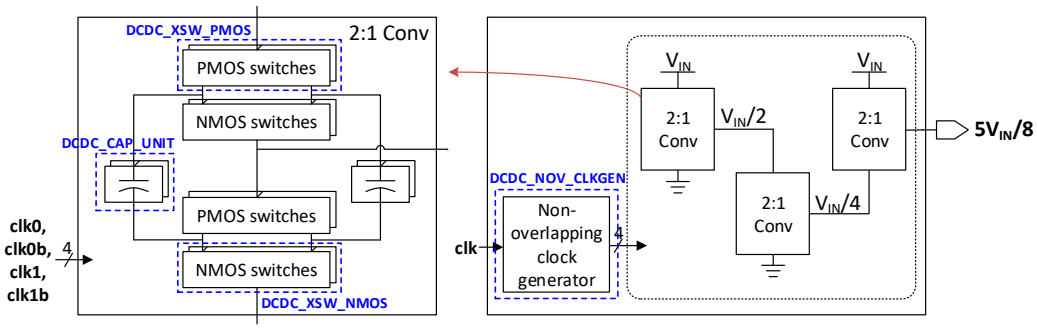

Fig. 9. Switched-Capacitor DC-DC converter architecture indicating the aux-cells

The generated switched-capacitor DC-DC converter (Fig. 9) utilizes the same base architecture as [26], which consists of multiple stages of 2:1 converters. The conversion ratio of the DC-DC converter is determined by the number of 2:1 converter stages and their configuration. The 2:1 converters are composed of 3 different aux-cells: DCDC_CAP_UNIT, DCDC_XSW_PMOS, and DCDC_XSW_NMOS. One additional aux-cell, i.e. DCDC_NOV_CLKGEN, is also required to generate two non-overlapping clock signals and their inverted signals for the 2:1 converters. Based on the input specifications (V_{IN}, V_{out}, I_{load}, and f_{clk}), the generator finds the optimal number of aux-cells in the 2:1 converters as well as determining the number of stages and their configuration.

4.6 SRAM

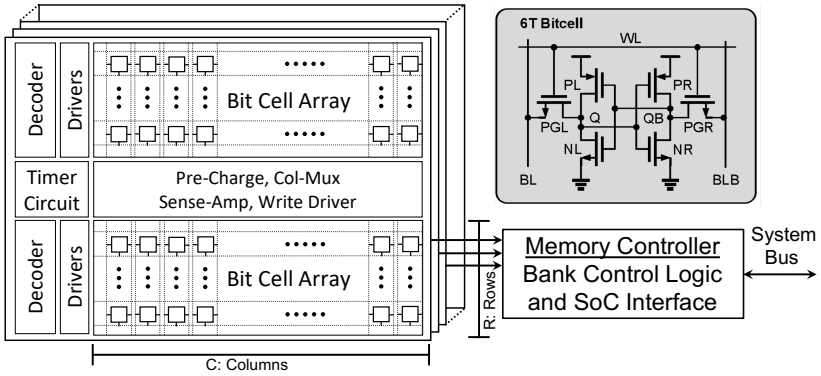

Fig. 10. SRAM architecture showing macros and bank strategy

A third party Commercial memory compilers (CMCs) [27–29] can generate an SRAM for a given PDK. However, they are the outcome of a human-driven design effort for each PDK and cover a fixed design space that usually emphasizes high performance. Such limitations restrict compilers' usage for applications such as ultra-low-power systems, which often operate in the nW to µW space. More importantly, the CMCs are not open source and may not be readily available due to cost or licensing issues, especially for newer technologies. Hence, a memory macro generation framework [30] is developed to address these issues and allow easy and autonomous generation of optimized memory macros in the design space where CMCs can not be used.

The memory generator creates fully-functional tapeout-ready integrated memories across a broad range of user specifications. The compiled SRAMs (Fig. 10) follow a standard multi-bank memory architecture. The memory generator uses a 6T bitcell, a row decoder, column mux, wordline driver, sense amplifier, write driver, and a pre-charge circuit as the aux-cells. The aux-cells are stitched together, bottom-up, to form a bank, and then a multi-bank memory. The user

input specifications are capacity, word size, operating voltage, and operating frequency. The generator adopts a hierarchical memory model to determine the optimal row and column periphery. The model helps to select the SRAM architecture and the leaf-level components that best satisfy the user specifications while minimizing energy consumption and delay.

5 SoC Generation

The top-level SoC generation begins with an iterative *SoC solver* to determine the optimal *composite design* which is a combination of blocks, analog specifications, and module connectivity. The strategy is guided by high-level user intent (i.e. target application and power/area budgets), available analog block generators, and a database of IPs. Analog generators are then invoked as necessary to generate bespoke blocks required to satisfy the specifications within the composite design. The generator outputs include all artifacts required to push the block through standard synthesis and APR tools. The outputs are also cached in an *IP database*, allowing for faster SoC generation if a matching entry already exists. Entries in the database can also be populated with 3rd party IPs such as processors and other peripherals.

The IP-XACT format is adopted to describe the composite design as well as the block designs stored in the database. Added Vendor extensions [31] capture additional analog data, simulation, and verification information. The *SoC integrator* begins by stitching the composite design together and translating it to its structural Verilog equivalent that can be run through digital simulation tools. The structural Verilog, along with all required artifacts from the database, is then passed through the embedded EDA tool flow to generate the final verified GDS. This same flow is pervasive across the framework and is also used by all generators (aux-cell, model, and analog). Tools within the flow cover all aspects of chip design including SPICE simulations, digital simulations, synthesis, APR, DRC, LVS, and extraction.

The rest of this section describes key components that make up the SoC generation stage.

5.1 SoC Solver

The primary task of the SoC solver is to derive a feasible solution to the supplied user intent and further derive an optimal solution that satisfies the intent. The user intent provides a basic sketch of the target application and includes minimal information relating to the performance, power, and area requirements. The solver takes an iterative approach to perform targeted design space exploration based on predictions learned from prior executions of the analog generators. It essentially builds a correlation between the user intent, analog block specifications (of all supported generators), and the generated block results.

Although the solver relies on a database of existing IP for faster iteration, it can quickly start from a cold cache to narrow down to an optimal solution. The

run-time of the solver is based on the number of blocks in the design, supplied budget, and how warm the cache is. The solver employs a heuristic algorithm to optimize the overall SoC metrics including power, performance, area, and other figures of merit.

The primary result of this process is a composite design in the form of a structural netlist. The netlist includes all specific module instances and information describing the ports and connectivity of all modules that constitute the design. The stitching process employs standard Arm AMBA protocols (e.g. APB and AHB) and includes all necessary interconnects and multiplexers.

5.2 IP-XACT and Database

The database entries are implemented using IP-XACT++, an extended version of IP-XACT. The added vendor extension catalogs supplemental meta-data relating to the IP. This non-tabular format allows for the storage of the PPA metrics as well as other figures of merits specific to each generator. This is in addition to the traditional IP information like ports, interfaces, memory maps, and validation data. The information associated with each generated instance of the IP allows the SoC Solver to quickly search the database for specific parameters and eases the stitching process to create the final netlist.

5.3 Embedded EDA Flow

The framework relies on an embedded EDA tool flow to accelerate the RTL-to-GDS process. It is a set of scripts and methodologies that leverage commercial EDA tools to accomplish the task of walking arbitrary designs through the synthesis and implementation process as quickly as possible. It builds on several modular abstractions to provide a PDK agnostic flow. Figure Fig. 11 shows the main abstractions inherent within the tool flow.

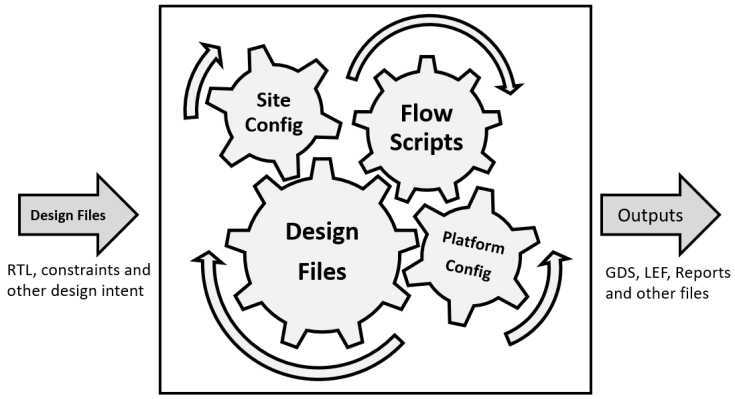

Fig. 11. Embedded EDA tool flow abstractions

Flow scripts: These are robust reference scripts for all the steps in the flow. It is collection of scripts span several EDA vendors used throughout the steps. It contains the *best practices* and recommendations from the tool providers and has been stitched together to form a generic end-to-end flow that can easily be customized based on several factors.
Platform Configuration: These constitute the PDK specific information, configurations, rules, and requirements for a specific technology node. There is a one-time effort to create this customization for newly supported technologies, however, subsequent designs can re-use that effort transparently
Site Configuration: These are configuration parameters that establish pointers to the required site-specific information. These are file paths to the PDK, standard cells, tool binaries, and license information for the EDA tools.
Design Files: This is the design-specific information that contains significantly less information about the EDA tools, PDK, and Site location since those have mostly been abstracted away and are expressed as customization to the generic flow in the EDA scripts. It is intended to express the actual design intent with little reference to the other abstractions.

Combined, these modular abstractions can be customized and combined together to form a block or chip specific flow. The various steps supported in the flow are simulation, synthesis, APR, LVS, DRC and extraction. It has support for 13 PDKs and has been validated with 6 different tape-outs across different PDKs. The flow is used at various levels and steps within the FASoC framework and has also been leveraged for other projects.

6 Evaluation

The framework has been fully verified in a planar 65nm and FinFET 12nm processes. The evaluation begins with a focus on the individual generators. The results presented explore the design-space possible with each generator and demonstrate full adherence to the user input specification in a 65nm process. Results are then presented from a prototype SoC created in 65nm process using this framework.

6.1 Analog Generation Results

Fig. 12 presents the results of several PLLs generated using different input specifications. It compares the input requirements against the simulated results after parasitic extraction. The results show that the generated frequency ranges cover that of the input requirements and with better phase noise levels. The highlighted PLL 8, corresponds to one of the PLLs integrated into the SoC prototype and also shows measured results that satisfy the given specifications

Fig. 13 shows the spice simulation results of multiple LDO designs after parasitic extraction. The graph shows the maximum load current at different input voltages corresponding to the input parameter array size for a dropout voltage of

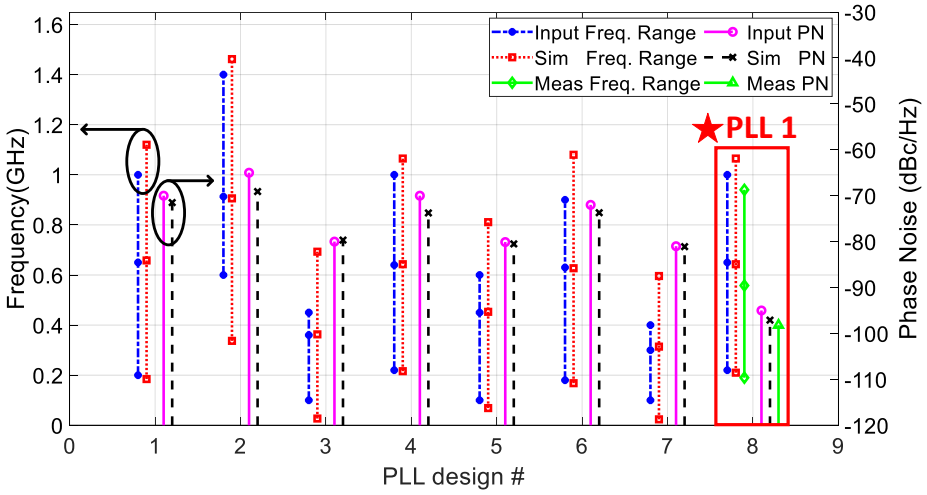

Fig. 12. Generated PLL designs for eight different input specifications. PLL1 is taped-out in the SoC prototype [1]

50mV. The highlighted measurements correspond to the input specification for blocks integrated into the SoC prototype with $V_{IN} = 1.3V$ and $V_{REG} = 1.2V$.

Fig. 14 presents the simulation results of various memory capacities across a broad range of architectural options and operating voltages (VDD). Each point on the curve corresponds to an energy-delay pair specific to an architecture (rows, columns, and banks) and VDD combination. The generator selects the Pareto-optimal design that satisfies the user requirements. The highlighted point on the 16KB curve corresponds to the memory block integrated into the SoC prototype.

Fig. 15 shows the spice simulation results of multiple temperature sensor designs after parasitic extraction.

Fig. 16 presents the simulation results of various numbers of V_{cm} switches. The generator selects the optimized value to satisfy the input specifications. By using a common-centroid placement strategy on the capacitors, the generator can also reduce the systematic mismatch which affects the accuracy of capacitance ratios. Table 1 shows the spice simulation results of cdl and pex netlists that closely match the input specifications with an area optimization.

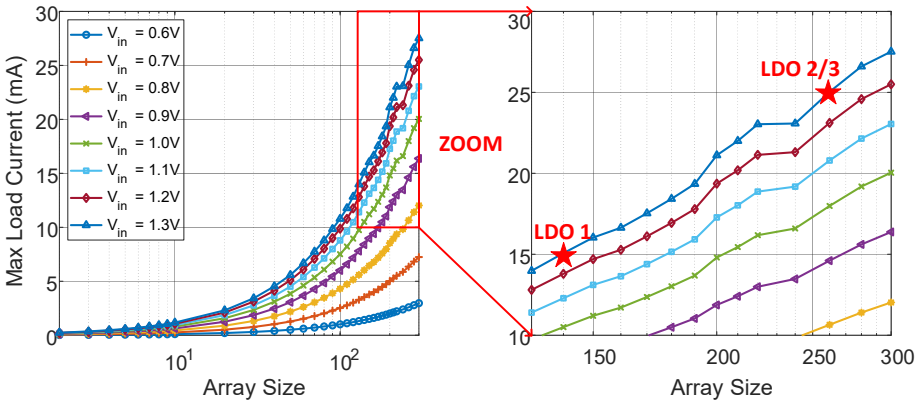

Fig. 13. $I_{load,max}$ vs. array size, for multiple LDO designs generated [1]

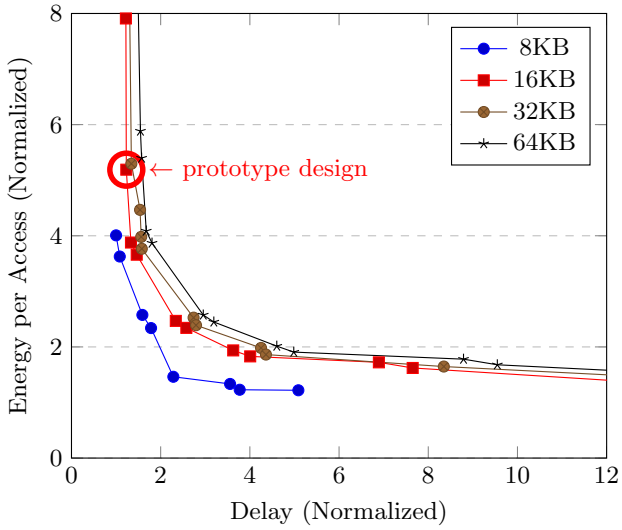

Fig. 14. Normalized energy and delay plots for various memory sizes while sweeping VDD.The results are normalized with respect to the 8KB memory. [1]

Table 1. ADC Simulation Results

Output Specifications	**CDL**	**PEX**
Sampling Freq (MHz)	1	
Unit Cap Value (fF)	2.6	
Area (mm^2)	-	0.04
Power dissipation (µW)	6.72	11.2
Effective Number of Bits	7.86	7.75

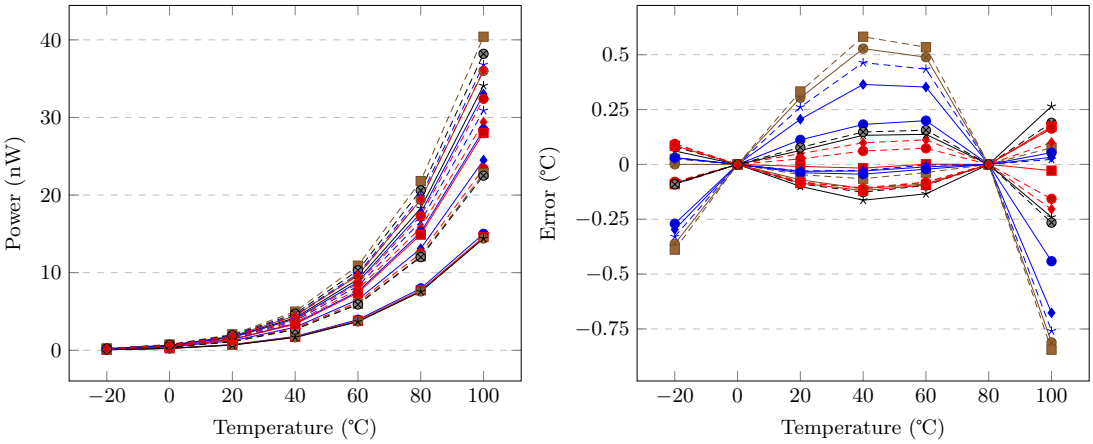

Fig. 15. Power and Error results against temperature for various temperature sensor designs (each fitted plot represents a unique design) [1]

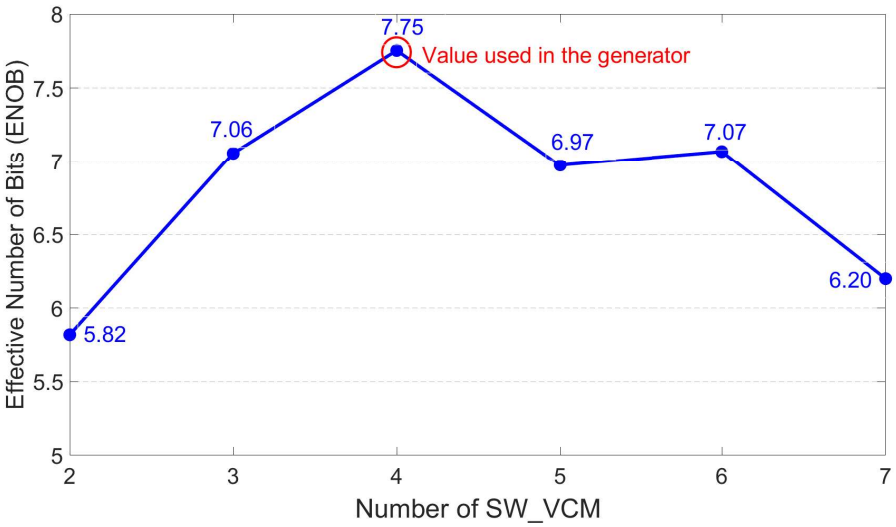

Fig. 16. Effective Number of Bits vs. Number of switches for V_{cm}

6.2 Prototype Chip Results

The 65nm prototype SoC design (Fig. 17) features 2 PLLs, 3 LDOs, a 16KB SRAM, and 2 temperature sensors fully integrated with an Arm® Cortex™-M0 in a 65nm CMOS process. Using off-chip connections, the entire SoC can be powered using one of the LDOs and clocked using the PLLs while monitoring the temperature of the chip.

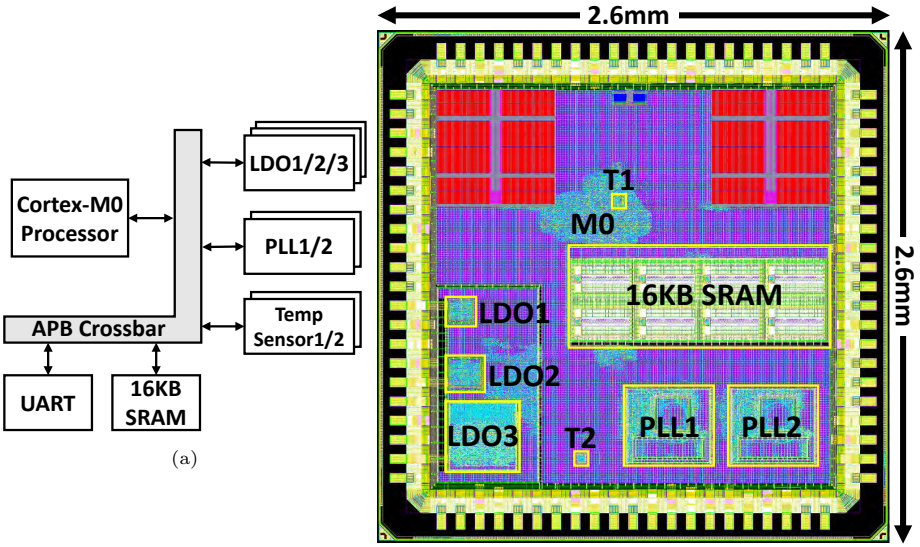

Fig. 17. Simplified block diagram (a) and annotated die photo (b) for the 65nm prototype SoC [1]

A similar 12nm prototype SoC (Fig. 18) features a PLL, 2 LDOs, a 64KB SRAM, 3 temperature sensors, a bluetooth transmitter, 2 SAR ADCs, a switched-capacitor DC-DC converter fully integrated with an Arm Cortex-M0 processor.

Fig. 12 presents results for 8 PLL designs generated from different input specifications, including one from the prototype, and the results show output performances in-line with the input specifications. The measured frequency is 10% slower while the phase noise matches the simulation and specification requirement. Table 2 summarizes the results for all PLLs in the prototype.

Table 3 shows the LDO $I_{load,max}$ measurements closely matching the input specification requirements. Compared to the comparator-based architecture (LDO1/2), the ADC based controller architecture (LDO3) achieves better transient performance with a 10x and 7x improvement in settling time and undershoot voltage respectively. The line and load regulation values are measured at

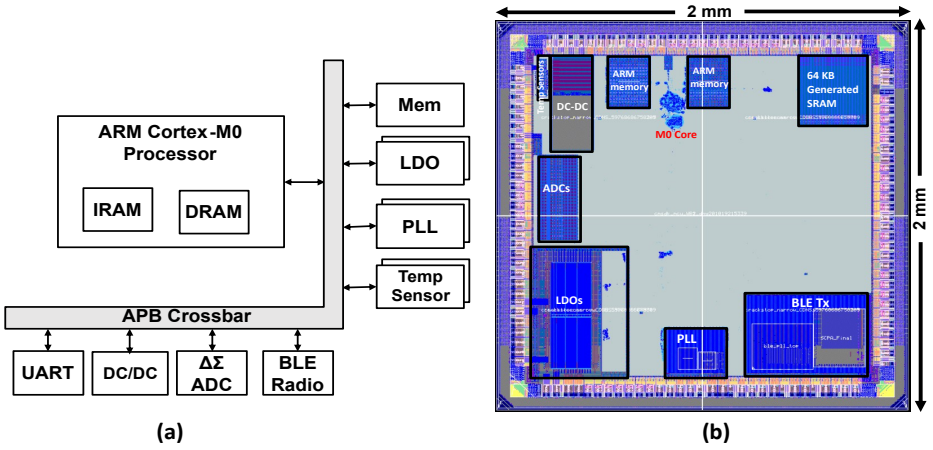

Fig. 18. Simplified block diagram (a) and annotated die photo (b) for the 12nm prototype SoC

Table 2. PLL Simulation vs Measurement Results [1]

Output Specifications	**PLL1**		**PLL2**	
	Sim	Meas	Sim	Meas
Min Freq (MHz)	200	190	170	150
Max Freq (MHz)	1,060	920	1,080	930
F_{nom} (Mhz)	643	558	627	548
Power@F_{nom} (mW)	7.20	6.90	8.06	7.70
Area (μm^2)	167,639.04		167,639.04	

Table 3. LDO Simulation vs Measurement Results @ 200MHz control clock [1]

Output Specifications	**LDO1**		**LDO2**		**LDO3**	
	Sim	Meas	Sim	Meas	Sim	Meas
Dropout Voltage (mV)	50	70	50	80	50	80
$I_{load,max}$ (mA)	15.00	15.38	25.00	24.84	25.00	23.72
Settling Time - Ts (µs)	1.1	1.8	2.1	2.9	0.12	0.19
Max Undershoot (V)	0.35	0.98	0.57	0.98	0.38	0.14
Max Current Eff. (%)	94.2	96.4	95.7	94.5	81.9	74.0
Load Regulation (mV/mA)	-	-1.00	-	-0.35	-	-3.6
Line Regulation (V/V)	-	0.180	-	0.004	-	0.950
Area (μm^2)	17,318.56		31,187.56		127,163.56	

V_{IN}=1.3V, V_{REF}=1.2V, and I_{load}=10mA. LDO3 load regulation is comparatively worse due to the high gain of the ADC based controller. While operating

at lower V_{REF} and I_{load} conditions, the line/load regulation degrades for all the LDOs because of the increase in relative switch strength.

The temperature sensor has an area of 2,620µm^2. A 2-pt calibration is performed at 0℃ and 80℃. Measured results show a sensing range between -20℃ and 100℃ with an accuracy of ±4℃.

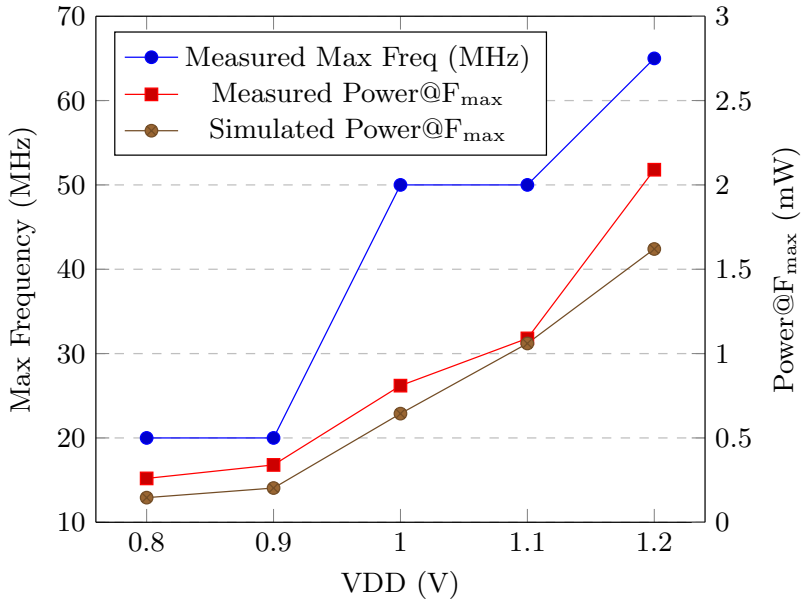

Fig. 19. Measured and simulated performance and power results of SRAM across VDD [1]

Fig. 19 summarizes the SRAM measured and simulated performance across the input operating voltage range of 0.8V to 1.2V. The SRAM peak performance is at 65MHz with the power consumption of 2.09mW at 1.2V, which exceeds the targeted frequency of 50MHz. The measured power for the SRAM also include the leakage power of the processor and peripheral interface. The generated SRAM has an area of 0.68mm^2 with the custom bitcell area occupying 0.4mm^2.

7 Conclusion

This chapter presented an autonomous framework that generates a completely integrated SoC design based on user input specifications. The framework is PDK agnostic and allows for faster turn-around times when building custom analog blocks and integrating them into larger SoC designs. The framework includes generators for PLL, LDO, temperature sensor, SAR ADC, switched-capacitor

DC-DC, and SRAM blocks. The framework can easily be extended to support more generators and different PDKs. The framework's validation was performed by creating and fabricating SoC prototypes in 12nm and 65nm processes. Silicon measurements for the analog blocks were inline with user requirements and simulation results. This work establishes a new milestone in creating a silicon compiler [32] that further reduces the complexity of realizing modern SoCs and cuts down on design time.

Acknowledgment

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7844. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. T. Ajayi, S. Kamineni, Y. K. Cherivirala, M. Fayazi, K. Kwon, M. Saligane, S. Gupta, C.-H. Chen, D. Sylvester, D. Blaauw, *et al.*, "An open-source framework for autonomous soc design with analog block generation," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 141–146, IEEE, 2020.
2. T. Ajayi, Y. Cherivirala, K. Kwon, S. Kamineni, M. Saligane, M. Fayazi, S. Gupta, C.-H. Chen, D. Sylvester, D. Blaauw, *et al.*, "Fully autonomous mixed signal soc design & layout generation platform," 2020.
3. Accellera, "IP-XACT - Accellera." https://www.accellera.org/downloads/standards/ip-xact. Last accessed 2020-05-03.
4. C.-Y. Wu, H. Graeb, and J. Hu, "A pre-search assisted ilp approach to analog integrated circuit routing," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 244–250, IEEE, 2015.
5. K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "ALIGN: Open-source analog layout automation from the ground up," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–4, 2019.
6. B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun, and D. Z. Pan, "MAGICAL: Toward fully automated analog ic layout leveraging human and machine intelligence," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2019.
7. A. Vladimirescu, R. Zlatanovici, and P. Jespers, "Analog circuit synthesis using standard eda tools," in *2006 IEEE International Symposium on Circuits and Systems*, pp. 4 pp.–, 2006.
8. Y. Park and D. D. Wentzloff, "An all-digital 12pj/pulse 3.1–6.0 ghz ir-uwb transmitter in 65nm cmos," in *2010 IEEE International Conference on Ultra-Wideband*, vol. 1, pp. 1–4, IEEE, 2010.
9. Y. Park and D. D. Wentzloff, "An all-digital pll synthesized from a digital standard cell library in 65nm cmos," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, IEEE, 2011.

10. E. Ansari and D. D. Wentzloff, "A 5mw 250ms/s 12-bit synthesized digital to analog converter," in *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pp. 1–4, IEEE, 2014.
11. S. Bang, A. Wang, B. Giridhar, D. Blaauw, and D. Sylvester, "A fully integrated successive-approximation switched-capacitor dc-dc converter with 31mv output voltage resolution," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 370–371, IEEE, 2013.
12. W. Jung, S. Jeong, S. Oh, D. Sylvester, and D. Blaauw, "A 0.7 pf-to-10nf fully digital capacitance-to-digital converter using iterative delay-chain discharge," in *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pp. 1–3, IEEE, 2015.
13. M. Shim, S. Jeong, P. Myers, S. Bang, C. Kim, D. Sylvester, D. Blaauw, and W. Jung, "An oscillator collapse-based comparator with application in a 74.1 db sndr, 20ks/s 15b sar adc," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pp. 1–2, IEEE, 2016.
14. S. Bang, W. Lim, C. Augustine, A. Malavasi, M. Khellah, J. Tschanz, and V. De, "A fully synthesizable distributed and scalable all-digital ldo in 10nm cmos," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 380–382, IEEE, 2020.
15. S. Kundu, L. Chai, K. Chandrashekar, S. Pellerano, and B. Carlton, "A self-calibrated 1.2-to-3.8 ghz 0.0052mm^2 synthesized fractional-n mdll using a 2b time-period comparator in 22nm finfet cmos," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 276–278, IEEE, 2020.
16. A. Rovinski, C. Zhao, K. Al-Hawaj, P. Gao, S. Xie, C. Torng, S. Davidson, A. Amarnath, L. Vega, B. Veluri, *et al.*, "A 1.4 ghz 695 giga risc-v inst/s 496-core manycore processor with mesh on-chip network and an all-digital synthesized pll in 16nm cmos," in *2019 Symposium on VLSI Circuits*, pp. C30–C31, IEEE, 2019.
17. C. Wolf, "Yosys open synthesis suite." http://www.clifford.at/yosys/. Last accessed 2020-05-08.
18. "Ngspice, the open source spice circuit simulator." http://ngspice.sourceforge.net/. Last accessed 2020-05-08.
19. S. N. Laboratories, "Xyce parallel electronic simulator (xyce)." https://xyce.sandia.gov/. Last accessed 2020-05-08.
20. D. M. Moore, T. Xanthopoulos, S. Meninger, and D. D. Wentzloff, "A 0.009 mm 2 wide-tuning range automatically placed-and-routed adpll in 14-nm finfet cmos," *IEEE Solid-State Circuits Letters*, vol. 1, no. 3, pp. 74–77, 2018.
21. M. H. Perrott, M. D. Trott, and C. G. Sodini, "A modeling approach for Σ-Δ fractional-N frequency synthesizers allowing straightforward noise analysis," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 8, pp. 1028–1038, 2002.
22. Y. Okuma, K. Ishida, Y. Ryu, X. Zhang, P.-H. Chen, K. Watanabe, M. Takamiya, and T. Sakurai, "0.5-v input digital ldo with 98.7% current efficiency and 2.7-μa quiescent current in 65nm cmos," in *IEEE Custom Integrated Circuits Conference 2010*, pp. 1–4, IEEE, 2010.
23. S. Weaver, B. Hershberg, P. Kurahashi, D. Knierim, and U.-K. Moon, "Stochastic flash analog-to-digital conversion," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 11, pp. 2825–2833, 2010.
24. M. Saligane, M. Khayatzadeh, Y. Zhang, S. Jeong, D. Blaauw, and D. Sylvester, "All-digital soc thermal sensor using on-chip high order temperature curvature correction," in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, IEEE, 2015.

25. S. Jeong, W. Jung, D. Jeon, O. Berenfeld, H. Oral, G. Kruger, D. Blaauw, and D. Sylvester, "A 120nw 8b sub-ranging sar adc with signal-dependent charge recycling for biomedical applications," in *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pp. C60–C61, 2015.
26. L. G. Salem and P. P. Mercier, "A recursive switched-capacitor dc-dc converter achieving $2^N - 1$ ratios with high efficiency over a wide output voltage range," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 12, pp. 2773–2787, 2014.
27. ARM, "Artisan memory compilers." https://developer.arm.com/ip-products/physical-ip/embedded-memory. Last accessed 2021-01-18.
28. Synopsys, "Designware memory compilers." https://www.synopsys.com/dw/ipdir.php?ds=dwc_sram_memory_compilers. Last accessed 2021-01-18.
29. D. Technology, "Memory products." http://dolphin-ic.com/memory-products.html. Last accessed 2021-01-18.
30. S. Kamineni, S. Gupta, and B. H. Calhoun, "Memgen: An open-source framework for autonomous generation of memory macros," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 3–2, IEEE, 2021.
31. R. Dreslinski, D. Wentzloff, M. Fayazi, K. Kwon, D. Blaauw, D. Sylvester, B. Calhoun, M. Coltella, and D. Urquhart, "Fully-autonomous soc synthesis using customizable cell-based synthesizable analog circuits," tech. rep., University of Michigan Ann Arbor United States, 2019.
32. D. Johannsen, "Bristle blocks: A silicon compiler," in *16th Design Automation Conference*, pp. 310–313, IEEE, 1979.