Programmable Analog System Benchmarks leading to Efficient Analog Computation Synthesis

JENNIFER HASLER AND CONG HAO, Georgia Institute of Technology, USA

This effort develops the first rich suite of analog & mixed-signal benchmark of various sizes and domains, intended for use with contemporary analog and mixed-signal designs and synthesis tools. Benchmarking enables analog-digital co-design exploration as well as extensive evaluation of analog synthesis tools and the generated analog/mixed-signal circuit or device. The goals of this effort are defining analog computation system benchmarks, developing the required concepts for higher-level analog & mixed-signal tools to utilize these benchmarks, and enabling future automated architectural design space exploration (DSE) to determine the best configurable architecture (e.g., a new FPAA) for a certain family of applications. The benchmarks comprise multiple levels of an **acoustic** , a **vision**, a **communications** , and an analog **filter** system that must be simultaneously satisfied for a complete system.

ACM Reference Format:
Jennifer Hasler and Cong Hao. 2023. Programmable Analog System Benchmarks leading to Efficient Analog Computation Synthesis. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 42 (August 2023), 23 pages. https://doi.org/XXXXXXX.XXXXXXX

Benchmarks serve as indicators of the fundamental computations perceived within a specific computing substrate. They allow new designers to envision potential applications of the technology and establish connections to desired user core applications. Benchmarks offer performance comparisons and facilitate optimization of the configurable technologies that embody the given technology. Utilizing multiple benchmarks is essential for implementing a technology on a configurable and programmable platform, ensuring that the technology is not optimized solely for a single problem.

Digital design has always had stable benchmarks. Digital high-performance computing benchmarks historically followed the solution of linear equations (**A x** = **b**) such as LINPAC [1] or related matrix operations (e.g. FFT, Neural Networks). Linear equations numerically match well with digital computation's (Fig. 1) large initial precision and moderate accumulation of partial results [2]. The resulting extensive digital system and tool benchmarks result in a stable landscape [1, 3–22]. A great collection of design benchmarks aim to to evaluate the

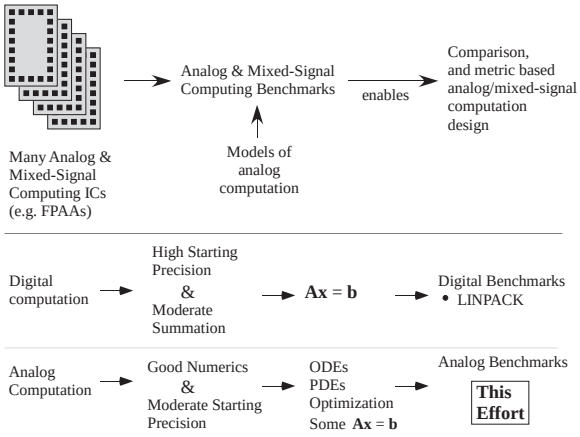


Fig. 1. Development of Analog and Mixed-Signal Benchmarks and resulting metrics. Benchmarks encapsulate what is currently understood of a particular technology.

Author's address: Jennifer Hasler and Cong Hao, ph67@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA, 30332-0250.

2476-1249/2023/8-ART42 $15.00
https://doi.org/XXXXXXX.XXXXXXX

hardware architecture, e.g., embedded processors, Field-programmable Gate Arrays (FPGAs), Graphic Processing Units (GPUs), and heterogeneous systems; representative examples include MediaBench [4], Rodinia [11], OpenDwarfs [12], SHOC [13], CHO [10], Rosetta [7], ERCBench [23], and Koios [8], which use OpenCL, high-level synthesis (HLS), or C/C++. Such designs cover a wide range of applications, such as arithmetic operations, image processing (encoder, decoder), cryptography (encryption, secure hash algorithms), processors (MIPS, RISC), signal processing (FFT, FIR), merge and sort, graph operations (BFS, DFS, MapReduce), wireless communication (error correction), vector and matrix multiplications, and the most recent machine learning workloads. In parallel, another category of benchmarks aims to evaluate computer-aided design (CAD) tools, e.g., high-level, logic, and physical synthesis [3, 5, 24, 25], such as CHStone [3] and MachSuite [5] for HLS, VPR benchmark suite for FPGA place and route [25], and EPFL benchmark for logic synthesis and optimization [24]. These digital benchmarks not only extensively evaluate hardware architectures and synthesis tools but also guide hardware-software co-design tradeoffs (e.g., [26–29]).

On the other hand, analog benchmarking has great challenges and has not been fully developed for decades. Analog and mixed-signal computing do not have significant benchmarks resulting from a long history of nearly no theoretical framework for computation for three important reasons. First, the *lack of automated analog design tools* compels most analog designers to heavily rely on graphics-based layout and simulation, demanding extensive manual effort with little generalization. Second, unlike digital design, analog designers must be *aware of underlying technology nodes* and available analog components. Analog designers often struggle with significant parameter variation, making it difficult to design high-level algorithms that do not expose lower-level analog components to users. Third, while digital benchmarks and circuits can have almost infinite variantions by reconfiguring parameters, *analog designs are typically case-by-case*; one reconfiguration (parameter change) may require a whole new design, largely limiting the scalability and generality of benchmarks. Therefore, a representative and rich analog benchmark suite has never been proposed or discussed.

Developments in analog and mixed-signal programmable and configurable systems, such as large-scale Field Programmable Analog Arrays (FPAA, e.g., [30]) and the SoC FPAAs (Fig. 2) [31], provided the opportunity for developing analog computing theory [2, 32–34] and created the need for synthesis tools and benchmarks that guide the tool development. Reconfigurable analog systems, like FPAAs, are a mixture of reprogrammable values that determine network topology as well as computation parameters. These devices have evolved without a set of benchmarks. Analog computing theory has been recently developed [2, 32–34], built on rich user experiences. Now that both the theory and reconfigurable FPAA platforms that implement these concepts are both mature, this paper defines the first analog benchmark suite.

This discussion aims to address these challenges based on the accumulated expertise in *analog reconfigurable devices* (e.g., [30, 36]) and *programmable circuit designs* (e.g., [37, 38]) by proposing the first representative and reconfigurable analog benchmark suite. The goal is to build an ecosystem of analog design and benchmarking by defining standard benchmark format and scope. This work formally defines the analog benchmark set at different levels, including application, algorithm, and block levels. New analog synthesis tools and hardware benchmark implementations, although currently being developed, will not be discussed as they are beyond the scope of this paper. The benchmark definitions can ease the effort of developing future analog synthesis tools by clearly defining the interfaces to analog libraries and by decoupling high-level synthesis and low-level (circuit) synthesis. The contributions and significance of this work are summarized as follows.

- This work proposes the first representative and rich set of *analog benchmarks* that are at different levels of complexity and are expressive with reconfigurable and customizable parameters.
- The proposed *analog benchmarks* arise heavily from analog computation capabilities. Analog benchmarks require understanding the fundamentals of analog computation to abstract circuit details into a meaningful top-down design flow.

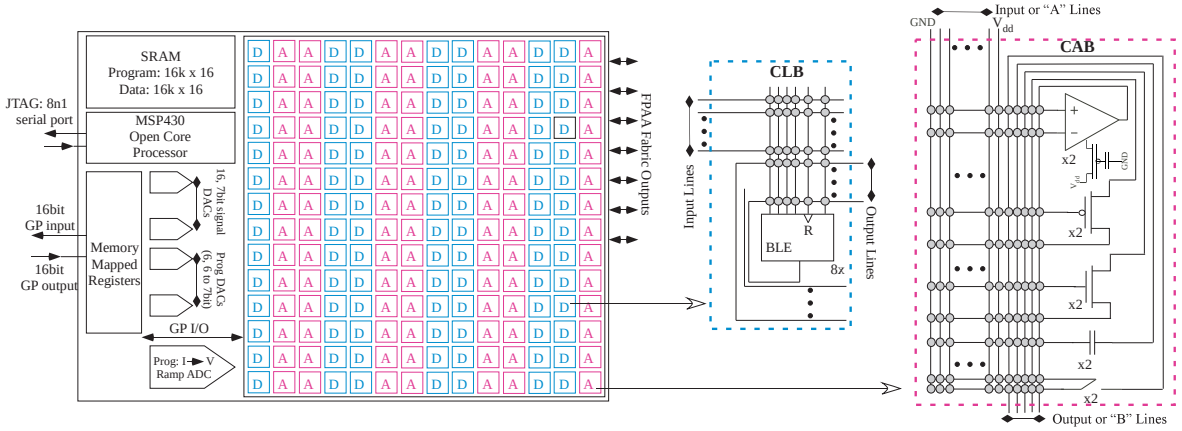


Fig. 2. SoC large-scale Field Programmable Analog Array (FPAA) computational blocks and routing architecture. The Computational Analog Blocks (CAB) are typical of earlier CAB designs, being combinations of transistors, OTAs, Floating-Gate (FG) transistors, OTAs, capacitors, T-gates, current mirrors, and signal-by-signal multipliers. The Computational Logic Blocks (CLB) are 8, 4 input Boolean Logic Element (BLE) lookup tables with latch. The SoC FPAA employs an open-source MSP 430 microprocessor (μP) [35] with on-chip structures for 7bit signal DACs, a ramp ADC, memory-mapped General Purpose (GP) IO, and related components.

- This work discusses the feasibility of the proposed benchmarks, demonstrating that they can be synthesized into existing analog circuits, either reconfigurable FPAAs or an integrated circuit (IC).

All designs, files, and reported numbers using these benchmarks are openly available[1]. We hope that the defined benchmarks in this work can provide standards for future analog synthesis (compilation) tool development, as well as further analog benchmark development, which requires not only effort from one group but from the entire community.

This discussion first overviews the current state of analog & mixed-signal systems and tools (Sec. 1). Then the discussion abstracts the capabilities of analog computing (Sec. 2) to show the resulting analog benchmarks (Sec. 3). These benchmarks are composed of an initial set of analog & mixed-signal *algorithm* blocks that provide solutions for these particular benchmarks (Sec. 4). Benchmarking (Sec. 5) also enables analog-digital co-design exploration and enables automated architectural design space exploration (DSE) to determine the best configurable architecture (e.g., a new FPAA) for a certain family of applications.

1 FPAA TECHNOLOGY AND TOOL OVERVIEW

The SoC FPAAs and earlier families of Floating-Gate (FG) enabled FPAAs demonstrated a number of core concepts, as elucidated from early large-scale FPAA techniques in 2002 [39] to today's devices (e.g., [30]). FPAAs have analog components with routing between analog and digital components, similar to FPGAs. Early programmable analog arrays [40–42] and early commercial devices (e.g., EPAC [43] or Anadigm [44]) were useful as *glue* logic and functions and for small occasional computations. Today's fully reconfigurable and nonvolatile FPAAs show considerable capabilities in ultra-low power computation (1000× lower than digital [45] following Mead's Hypothesis [46]), signal processing, and embedded machine learning (ML) [30] from ICs fabricated in a 350nm CMOS process. Programmable analog FG techniques have demonstrated precision components [47] built on work enabling programmable design for temperature [48]. FPAAs have also been integrated with sensors, possibly fabricated on the FPAA fabric [49, 50].

[1] https://hasler.ece.gatech.edu/AnalogBenchmarks

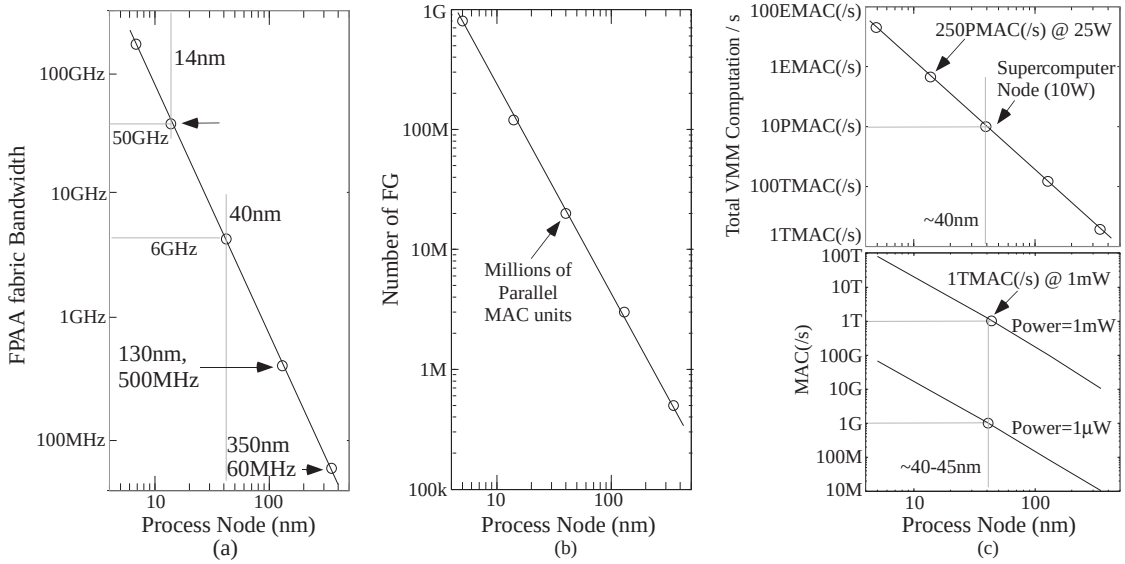


Fig. 3. The potential of scaled down FPAA devices, enabled through synthesis tools, from extrapolations from experimental measurements and early studies of scaled down FG and FPAA devices (350nm to 5nm). *These opportunities motivate the need for synthesis tools to enable these new devices at commercial timescales.* (a) scaling of the resulting fabric computational bandwidth. (b) scaling of available analog FG devices, and therefore the number of computational parameters and devices. (c) The computational and computational efficiency opportunities for scaled down 5×5mm^2 FPAA devices. Existing FPAA systems (e.g. 350nm CMOS) enable significant computation (≈ 1 TMAC(/s), MAC = Multiply ACcumulate) and computation at low energy levels (10GMAC(/s) in 1mW), scaled down FPAA devices enable embedded supercomputing capabilities in mW power levels.

Large-scale programmability is essential for these FPAA devices, typically instantiated using analog-programmable FG devices available in standard CMOS for the memory and routing elements. Today's SoC FPAA [31] utilizes 600,000 programmable parameters in 350nm CMOS (Fig. 2 and 3). Analog programming of the single FG pFET device fabric enables computation in routing fabric as well as Computational Analog Block (CAB) elements [51]. FG parameters tend to provide a 1000× advantage over other Si memory approaches at roughly 8-10bit level [30]. FG devices have demonstrated long-term (10-year) lifetime across multiple IC processes from 2μm to 40nm linewidths [47, 52, 53] with precision (e.g., 14-bit) targeted (re)programming of heterogeneous arrangements of FG devices [54]; FG devices enable the direct elimination of mismatches or the setting of desired target values in the configurable structure. FPAA devices are capable of secure operation [55].

From experimental measurements in scaled down processes (e.g. 14nm or 40nm, [56]), tomorrow's FPAAs [36] will use parallel operations available in energy-constrained environments (Fig. 3), enabling ubiquitous devices for embedded *and* high-performance applications. Programmable techniques have shown promise enabling scaled down analog design, avoiding device mismatch difficulties in scaled process analog IC design [53, 56]. Besides higher density and lower energy consumption, smaller CMOS linewidths (e.g., 40nm) enable RF frequency signals (e.g. 4GHz and higher) through FG routing fabric [53]. Current mixed-signal systems and designs are highly restrictive of IC process choices, due to the manual design effort involved when moving to a new IC node but not technical limitations.

The large system size makes design tools essential for FPAA system development, similar to FPGAs, and the development of FPAA devices has been the motivation for the most advanced efforts in analog tool development. FPAA design tool development [57–59] gives users an increased ability to create, model, and simulate analog and digital designs. High-level design tools implemented in Scilab/Xcos can automate the circuit compilation

into a *switch list*, the description of the programmed FPAA hardware [57]. These graphical tools correspond to the natural analog dataflow computation, enabling a non-circuits expert, like a system application engineer, to investigate particular algorithms. These tools also enable (Fig. 4) system level design and macromodeled simulation (level=1) and circuit level design (level=2) [59], including both FPAA targeting and simulation [60]. The chip details are specified in architecture files for the analog-digital SoC FPAA, enabling place and route [61] in a user transparent process.

Synthesis tools can unlock the potential of analog architectures to achieve real-time computation, signal processing, and inference and learning for low SWaP (Size, Weight, and Power) systems in commercial timescales (Fig. 3). Empowered by the initial FPAA design and synthesis tools and recent developments for analog & mixed-signal standard cell library formulation and implementation [62, 63], opportunities exist for synthesis from text-based behavioral (e.g. Python) and high-level (e.g. Verilog) abstractions to configurable devices, custom silicon IC layout (.gds), or the design of new FPAA devices, as well as macromodeled simulation from these high-level formulations[2].

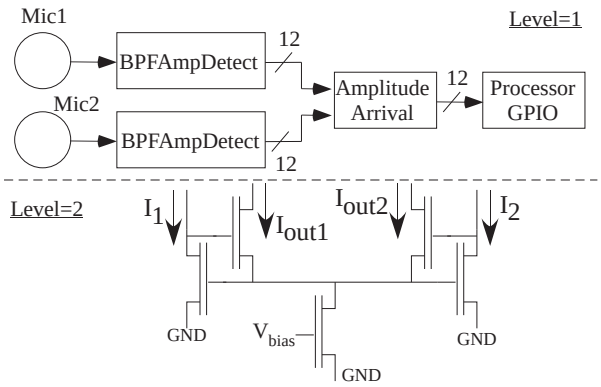


Fig. 4. Review of the two levels of analog abstraction, level=1 for vectorized, voltage-mode, system level dataflow representations, and level=2 for full current-voltage analog cell representation used by an analog designer.

2 BASICS OF ANALOG COMPUTING

Configurable FPAA devices motivate the need for analog & mixed-signal benchmarks in using and evaluating automated synthesis tools. Formulating these benchmarks require understanding the nature of analog computation. The digital VLSI revolution [64] with IC process scaling [65–67] is because of the well-developed digital theoretical foundations [68] and frameworks. However, analog computation largely relies on artistic development, and therefore could not directly take advantage of IC process scaling. The following four decades have seen the slow building of analog computing concepts.

The initial FPAA successes made analog computing theory both a practical direction with realistic technology considerations, as well as a necessary development for system design. Analog computation nowadays has far exceeded its early state of a few passive components around some op-amp blocks and goes beyond a few signal conditioning circuits using resistors around an op-amp or a useful component used in digital infrastructure (e.g., PLL). Therefore, widespread programmability is essential for any computing approach and analog component design must avoid being stuck in the artistic space where users need to be developing designs that fit well with the larger system technology.

The development of analog abstraction and the design of modular analog computing components directly come out of the FPAA tool development and enable the opportunities for benchmark systems [32]. The development of analog numerical analysis dispels the belief that analog is an imprecise computing medium compared to pristine digital methods, but rather, digital and analog computations have their own strengths in corresponding areas [2]. The development of analog architectures gives a roadmap for efficient analog computation, showing that communication is the primary cost in analog computing given the local analog computing efficiency,

[2]Multiple papers are currently in submission, and upon acceptance will be referenced in this section. We are enthusiastic to provide accepted papers to the reviewers.

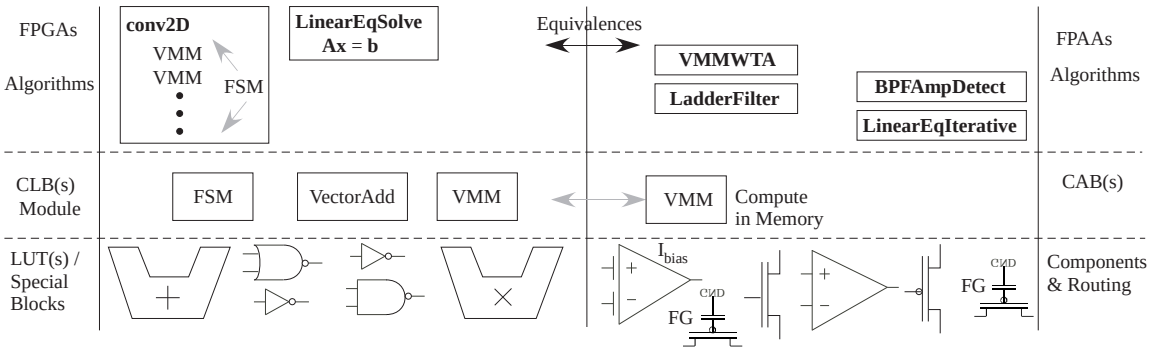


Fig. 5. Parallel FPGA and FPAA Levels of complexity. As FPGAs have algorithms that are composed of modules utilizing one or multiple Computational Logic Blocks (CLB) that in turn utilize fundamental logic gates (e.g. LUTs) and components, FPAAs have algorithms that are composed of blocks utilizing one or multiple CABs that in turn utilize multiple components (e.g. transistors or Transconductance Amplifiers (TA)) or routing elements.

particularly communication with memory components [33]. These three parts enabled the development of a real-valued Turing Machine model for analog computation, providing a theoretical framework for these approaches [34]. The real-valued computation and theoretical modeling reveal the limitations of digital (integer-valued) simulation of analog computation: digital simulation is not, and will not ever, be completely sufficient for analog system design compared to physical system implementations. These concepts begin to make analog computation a top-down design methodology.

Analog Computation. Analog computation directly utilizes Ordinary Differential Equation (ODE) or Partial Differential Equation (PDE) computations in vector or matrix formulations as a result of the ideal summation (enables long summations) and continuous-time integration properties [2]. Both analog and digital computations compute Vector-Matrix Multiplications (VMM) in an efficient form given their technology framework, where analog VMM tends to be 1000× more energy efficient than digital (e.g. [45, 69]). The extension of the dense mesh crossbars of FG elements for VMM computation started the Computing-in-Memory (CIM) efforts in 2001 [38]. ODEs describe continuous-time physical systems, such as electronic circuits, governed by physical laws. The integration of capacitors and currents is a physical law with no inaccuracies [2]; the primary device mismatch is threshold voltage mismatch that is directly addressed through FG programming. While a digital problem usually can be solved by simplifying to $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, an analog problem can be solved by simplifying it to a set of ODEs. An analog system can solve systems of linear equations by setting up a differential equation with a steady-stated described by $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ [70]), although an effective analog $\mathbf{L}\,\mathbf{U}$ decomposition still remains an open question. Although the ODEs for these benchmark problems can be digitally simulated, building the full system can enable complete characterization and thus is beneficial.

Analog Abstraction. Analog computation follows a flow-graph representation (e.g. Simulink or Xcos) effectively assuming single-ended voltage-mode busses of signals. The levels of analog components have similar parallels to digital components, where the point of connection and direct comparison is at the algorithm level (Fig. 5). These vectorized (Fig. 6) signal busses result in efficient high-level graphical representations. These analog system level designs are built from abstracted level=1 blocks (Fig. 4), enabling a dataflow graph of macro-modelled single-ended vectorized voltage input and output signals [32, 57, 59]. The capabilities to do classical analog design are continued and enabled through level=2 blocks that utilize the fullness of the current-voltage implicit functionality of the available devices (Fig. 4). These experimentally demonstrated level=2 devices would be the essential building blocks for new level=1 devices when fully macromodelled and vectorized. The resulting system in the FPAA or in the resulting structure will require known blocks for control flow and programmable infrastructure components.

Analog Architecture. The proper analog or digital architecture [33] choices directly impact the design of an efficient and high-performance system. For sensor systems, one wants an *end-to-end* computation from analog input signals from a sensor interface to refined digital or near digital classified outputs. Analog computation ideally operates at the speed of the input or output data, since buffering (e.g. memories) is expensive due to the communication cost. Effectively, the computation becomes free and communication is the primary issue, arguing for co-located memory and computation. Other approaches, such as using an analog computation as a specialized routine (or co-processor) to a main digital system, often are a mismatch to the technology (e.g. requiring many DACs and ADCs at the boundary), to the numerics, as well as to the targeted application problem (e.g. Mythic Semi [71]).

3 DEFINITION OF ANALOG BENCHMARKS

The proposed benchmark suite addresses a wide range of analog and mixed-signal computations (Fig. 7). The development of programmable and configurable systems, whether digital or analog, requires multiple benchmarks not to optimize for a single computation but to enable wide applicability of the technology. This benchmark suite could also be solved using digital computation when including the necessary data converters, providing useful comparisons between digital and analog designs and eventually co-design opportunities.

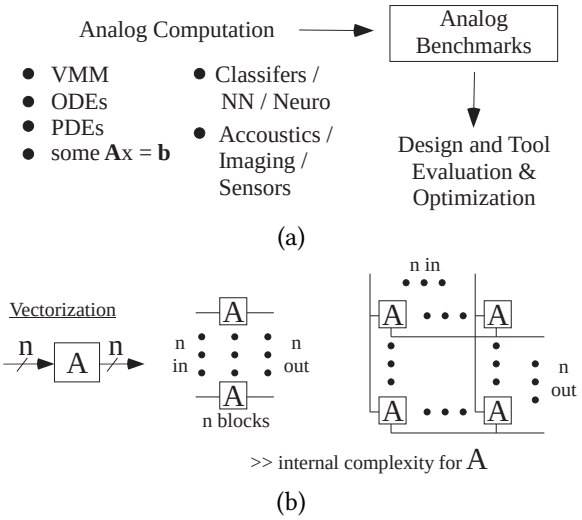


Fig. 6. Development of Analog and Mixed-Signal Benchmarks and resulting metrics. (a) Benchmarks encapsulate what is currently understood of a particular technology. Analog Benchmarks build from the advantageous analog computations. (b) Analog abstraction requires encapsulating analog computations into a dataflow framework, requiring a range of vectorized analog components.

Analog benchmarks must consider a wide range of analog computations from sensor signals, such as sound, image, or communication. Benchmarks are end-to-end definitions from sensors to refined computation output, providing a fair comparison of analog computing approaches with other technologies. Fig. 7 gives an overview of our proposed benchmarks spanning a wide application range with different parameters and sizes (Case I to Case IV). There is one significant benchmark in speech processing for **acoustic** processing (Sec. 3.1), one significant traditional analog system design (a 10^{th} order programmable Continuous-Time (CT) *filter*, Sec. 3.4), one significant benchmark in **vision** that includes image classification (Sec. 3.2), and one significant benchmark in **communications** (e.g. RF) computation (Sec. 3.3). Two of these benchmarks (acoustic and vision) utilize embedded ML as part of their operation. The communication benchmark likely will use ML, and the **filter** benchmark could be framed as an optimization problem solved through ML. The labeled training and testing set applied to the sensor input specifies the benchmark inputs and outputs. Part of the benchmark examples requires training the benchmark parameters for the analog implementation. The training can be done either off-line or on-the-fly on the physical system; on-chip learning has already been demonstrated for analog computation, including configurable FPAA systems [72, 73]. One expects further benchmarks to be defined along these directions.

These benchmarks (Case I to Case IV) are designed for multiple problem sizes (Fig. 7) to illustrate as well as normalize between different IC technology nodes. One might find certain problem sizes that are not achievable

		Case I		Case II		Case III		Case IV	
	Computation	Size	Freq	Size	Freq	Size	Freq	Size	Freq
Acoustic	Command / Speech recognition								
	from microphone signal:	4[a]	20kHz	32[b]	20kHz	256	20kHz	1000	20kHz
Vision	Image Classification	MNIST[c]	60fps	MNIST[d]	60fps	1Mpixel[e]	60fps	10Mpixel[e]	60fps
Comm	Beamforming and								
	Demodulation (0.01× input freq)	8	3MHz	16	50MHz	32	2GHz	64	20GHz
Filter	10^{th} Order Programmable LPF	1	300kHz	1	3MHz	1	30MHz	1	300MHz

[a] Command Word Classification, [b] Small Vocabulary Classification

[c] Compressed DCT, 19-8-6-4 NN, [d] 3-layer NN, [e] Image convolution / classification

Fig. 7. Creating the first Analog & Mixed-Signal Benchmarks. Defining the first benchmarks for analog and mixed-signal computation requires encapsulating core analog computations. Analog computation centers around solving differential equations using sensor inputs. These starting benchmarks can have additional benchmarks added as technology continues to grow.

by a 350nm CMOS IC process, and yet might be achieved by a 130nm, 40nm, or 14nm CMOS IC process. Reconfigurable applications directly connect to the roadmap of what can be built in different CMOS linewidths, and the proposed benchmarks provide system-level computational comparisons. All four benchmarks in a given case must be created and characterized when evaluating a reconfigurable platform or analog synthesis tool. Cases I through IV vary in levels of difficulty and complexity, showing different capabilities for a system satisfying all four benchmarks. For example, the SoC FPAA (350nm CMOS) would handle the Case I benchmarks (e.g., [31, 73–75]), while further improvements could address some of the Case II benchmarks. Scaled CMOS processes (e.g., 40nm CMOS) are capable of reaching the Case III and Case IV benchmarks (e.g., [53], Fig. 3). Potentially, benchmarks in more complicated cases (e.g., Case V) can be defined on top of the success of earlier benchmarks. The important benchmark measurements would be energy and/or power consumption, accuracy, and the area or resource utilization for the computation. Another measurement (Imager Benchmark) identifies the amount of required non-local communication, as well as intermediate memory storage, to minimize architectural costs when operating at the input (sensor-driven) and output (classification/computation). Each benchmark uniquely stresses the analog programming capability of the resulting analog system. The particular physical solution will depend on the available sensors (e.g. types of imagers, types of microphones) to the engineers.

Analog circuit problems outside of these computing spaces would not make good benchmarks, because they are often highly specialized analog design questions. For example, level-shifting (e.g., voltage) amplifiers that are often solved through specialized technologies (e.g., 100V supply GaN amplifiers) are not a useful benchmark; when needed, such devices can be integrated into modules with a programmable device. Similarly, specialized sensor interface blocks or data converters might be used in a benchmark system but are not considered as a benchmark. Other traditional analog circuit blocks that assist in digital systems, such as Phase Locked Loops (PLL), are typically blocks in a larger system (e.g., a standard-cell block), and thus are not a full benchmark either. Analog computation benchmarks should focus on larger analog computing capabilities.

3.1 Acoustic benchmark: Microphone to Word Classification

The first benchmark is a programmable end-to-end microphone-to-symbol keyword/word/acoustic symbol classifier. The input bandwidth is 20kHz for all problems, set by the input microphone sensors in the range of human hearing. Existing solutions use a filterbank of incoming signals in a near exponential spacing of center frequencies (e.g., 50Hz to 10kHz) that model the human cochlea dynamics (e.g., [31], preserving the highest signal & frequency information while refining the data representation. Current solutions might take the raw microphone input through a set of programmable parallel bandpass filters (16-24) followed by an amplitude detection (Fig. 8). The amplitude detection produces a set of analog signals indicating a filtered magnitude of a set of frequencies.

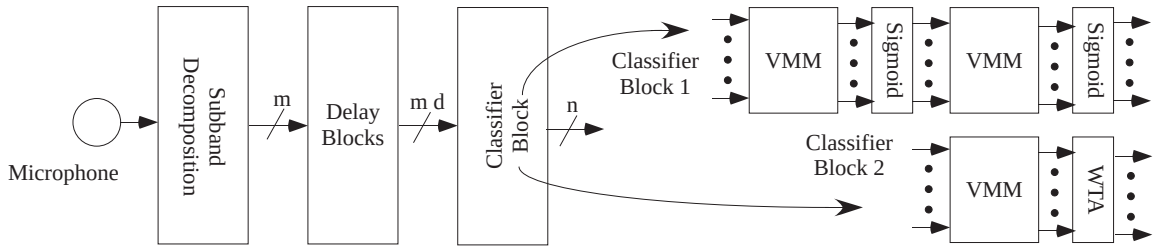


Fig. 8. Benchmark circuit for the command-word recognition benchmark (acoustic). Multiple existing physical implementations for the classifier block would include a VMM+WTA universal approximator classifier.

Some existing solutions implement additional dynamics or approximate delays [75] before or in the classifier. Programmability enables repeatable front-end computing structures by eliminating the effect of mismatch, resulting in a near-ideal transfer function for each filter channel.

This benchmark classifies the sound signal representations into words or other acoustic symbols (Fig. 8). Typical implementations of such classifiers use a feedforward computation, although recurrent architecture might present significant performance improvement for different vocabularies with little additional system cost to implement the resulting ODEs. The classifier (Fig. 8) could be implemented as a multi-layer neural network (NN) (Fig. 9), or one or multiple layers comprised of VMM + a Winner-Take-All (WTA) universal approximator/classifier. The quality of the programmability available for an experimental solution will directly impact the measured metrics to this benchmark. The output is a binary digital classification, or a low-bit (2-4bit, straightforward ADC) sparse representation that gives a confidence measure for the identified words. The sparse output has a low sample rate for the next system (e.g., 1-10 samples/s) that likely would be event encoded. The classified vocabulary size (Case I: 4 Case II: 32 Case III: 128 Case IV: 1000) shows the system classification capability. The labeled input datasets for this benchmark are:

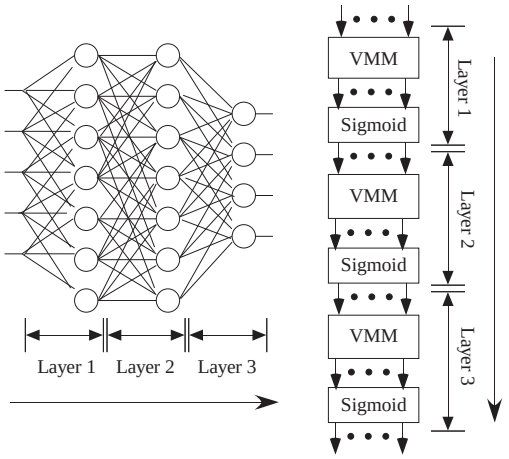


Fig. 9. Embedded neural networks are essential for most of the benchmark problems, where a number of potential architectures utilizing one to multiple layers might be utilized.

- Case I: 4 words from TIMIT [76] or TI digits database [77].
- Case II: Speech Commands Dataset [78]: 32 labeled words from multiple (5) speakers (made available through Google)
- Case III: 128 words from the labeled dataset LibriSpeech [79] (similar sized datasets are possible)
- Case IV: 1000 words from the labeled dataset LibriSpeech (similar-sized datasets are possible)

One can find openly available versions of all these datasets. Forms of this benchmark (primarily Case I) have been demonstrated within configurable platforms [31, 73, 80]. As one might only be able to obtain a microphone with an LNA, any reported result must describe if the system uses a raw microphone or microphone with LNA as well as the input signal SNR levels. The classified output is defined through the labeled data set. The users of a particular dataset will have to train their network for the particular dataset, e.g., through offline training. If users have technology capable of on-platform training, that should also be identified with the resulting energy efficiency of the training. Metrics for this benchmark include energy efficiency, accuracy of classification starting from sensor input, and area of the resulting system.

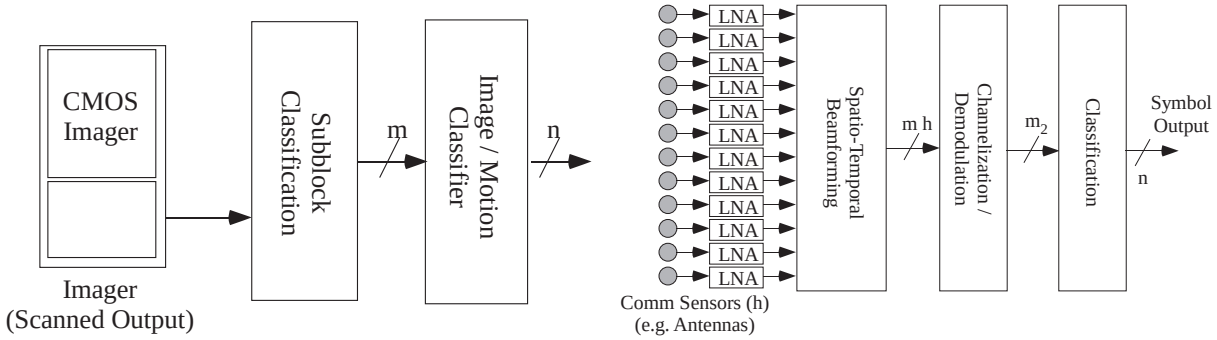


Fig. 10. The vision and communication Benchmark circuits. (a) Vision benchmark circuit. (b) Communication benchmark circuit.

3.2 Vision benchmark: CMOS imager to Classification

The large number (e.g. millions) of pixel sensors for vision classification challenges the analog computational and communication complexity. The size of the image classification problem becomes the primary parameter, where the frame rate tends to be nearly fixed for most applications looking for human-like performance. The classifier should operate at the input signal rate. The system must start from a commercially available (e.g. production or R&D moving to production) CMOS imager requiring an input signal that is a two-dimensional scan of the image plane or an event representation of the image plane (Fig. 10). This communication bottleneck significantly constrains the computing architecture as one must minimize the movement of data into short-term storage [33]. Many computations utilizing two-dimensional representations experience similar issues. Commercial imagers have either digital or analog datapath out of the imager; having either input signal into the analog processing is reasonable, even if an initial DAC is required to recover the single analog signal. Communication from the imager is part of the benchmark measurements. An imager die stacked on a configurable analog structure enables a number of different computational opportunities [36], although often with significant additional cost. The first two vision processing Cases (I and II) are static image classification (Fig. 9). The last two vision processing Cases involve both image classification as well as motion processing (changes between images), utilizing recurrence and dynamics in analog computation. The particular classifier builds up representations (Fig. 9), starting from symbols from initial local image refinement for the larger image processing. The input would need to be projected on an imager module that must be large enough for the entire image and then processed for the full benchmark problem to the classified outputs. Identifying the imager device (e.g. commercial product, event imager, specially built device) allows for comparison of the resulting algorithm. The labeled datasets form the inputs and resulting desired outputs for the benchmark; the labeled datasets would be:

- Case I and II: MNIST image dataset [81]
- Case II: COCO 2017 dataset [82], the dataset used in the ResNet [83] classification models
- Case III & IV: SVD Video labeled dataset [84]

A particular labeled dataset requires training the physical computing system for that dataset as classifier implementations are design dependent. Metrics for this benchmark include energy efficiency (continuous-image classification), accuracy of classification starting from sensor input, and area of the resulting system.

3.3 Communication benchmark: Beamforming and Classification

The communication benchmark is a common module for RF-based communication systems from sensors (e.g. Antenna structure) through classified output. Fundamental computations include spatio-temporal filtering for beamforming and frequency shaping (e.g. equalization) & filtering, potential channelization (linearly spaced

bandpass filters), demodulation, and classification of the resulting complex baseband signals. The spatio-temporal filtering requires approximate delay lines or similar decompositions and a VMM (e.g. [75]). Symbol classification (Fig. 9) includes identifying the complex transmitted value in say a QAM64 or higher standard. The cases are scaled by center bandwidth frequency, as a higher operating frequency significantly increases the system complexity (Case I: 3MHz Case II: 50MHz Case III: 2GHz Case IV: 20GHz). The cases are scaled by the number of input signals for a single element for the front-end spatio-temporal filtering (Case I: 8 Case II: 16 Case III: 32 Case IV: 64). The resulting demodulated output then gets classified into symbols. The number of labeled communication (e.g. RF) datasets is limited, although a few exist (e.g. [85]) to be used for this measurement. It is assumed that the output signals would be coming out of a communication source (e.g. Antenna + LNA) and should be identified. Metrics for this benchmark include energy efficiency, accuracy of classification starting from multiple voltage-signal inputs, and area of the resulting system.

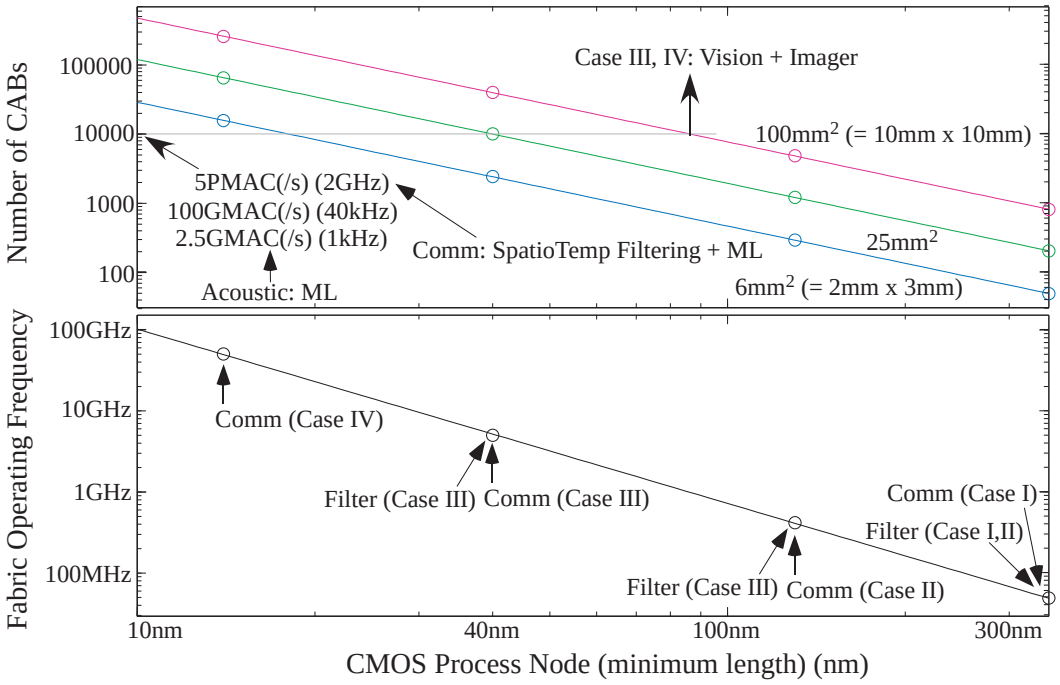


Fig. 11. Tradeoffs between FPAA implementations and benchmark problems for particular cases (Case I through Case IV). *Upper Graph*: Complexity for 2x3mm^2, 5x5mm^2, and 10x10mm^2 die sizes as a function of process node, calling out the 350nm, 130nm, 40nm, and 14nm CMOS nodes. Fabric operating frequency *Lower Graph*: Fabric operating frequency (frequency to near DC) as a function of process node, calling out the 350nm, 130nm, 40nm, and 14nm CMOS nodes.

3.4 Analog benchmark: Low Power, High-SNR Programmable Filtering

The analog computation functions are balanced by a *traditional* analog computation problem of a programmable low-power and high SNR 10^{th}-order Low-Pass Filter (LPF). A BPF is already addressed through the acoustic benchmark, therefore, this effort focuses on programmable LPF filters. Existing solutions typically utilize a cascade of 2^{nd}-order programmable filter blocks [86] or a programmable ladder filter block [75]. This benchmark would be measured by the consumed resources/area, circuit power, and achieved SNR. This benchmark will stress the programming accuracy and capability of the solution; all circuits required for programming are addressed and included in this benchmark.

The target frequency is the highest of all possible frequencies, although the filter needs to also operate with 10× and 100× lower frequencies than the maximum case frequency with the same specifications as a demonstration of the system's programmability. The frequency range is the filter bandwidth from DC to near DC to the frequency of interest for the passband region with a passband ripple less than 2% (0.2dB) with the stopband 1.2× the passband frequency at a factor of at least 100 (40dB) lower than the passband gain. The test input into this system would be a sinusoid measurement where the sinusoidal frequency is at the place of the highest distortion (typically near the passband frequency edge) at the highest filter input amplitude. The output measurement is a spectrum of the resulting output, illustrating the primary frequency amplitude, the harmonic amplitudes, and the noise amplitude. From these measurements, one obtains the noise floor, Signal-to-Noise Ratio (SNR), maximum input amplitude, and harmonic distortion, as well as the resulting power and area consumed. These typically achievable measures will provide a sufficient test of the resulting analog filter design capabilities of the technology [87–89] as well as the traditional analog circuit capabilities.

3.5 Analog Computation Design pushed through Simultaneous Benchmarks

These benchmarks simultaneously push different characteristics important for reconfigurable computational analog design as well as the tools for synthesizing a range of analog designs (Fig. 11). Many opportunities are possible with existing SoC FPAA devices (350nm CMOS), and yet, each benchmark problem shows particular design choices in an architecture exploration process. All four benchmarks must be satisfied by a particular configurable system or synthesis tool, each at their particular Case level. Custom designs will have some similar tradeoffs when targeted and will show design tool capabilities.

The *communication* benchmark primarily pushes the fabric frequency limitation (Fig. 11). The size of the spatio-temporal filter for Case IV is possible in one to four SoC FPAA devices, although the frequency limitations around 50-60MHz limit this benchmark. The Case III frequency metric (2GHz) and spatio-temporal filter complexity was already exceeded by a 2mm × 3mm 40nm CMOS FPAA [53]. The Case IV metric should be easily satisfied at the 14nm CMOS node.

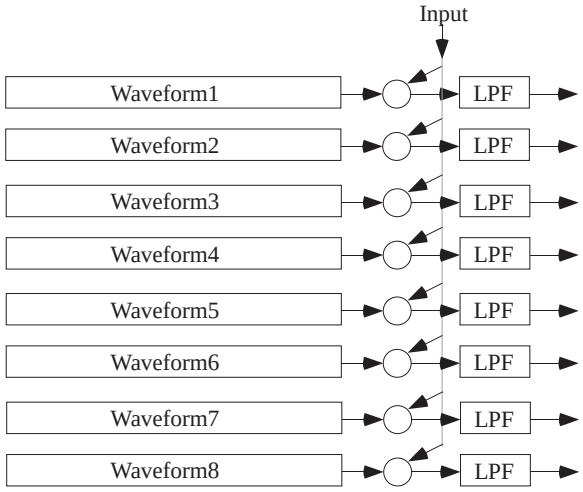


Fig. 12. Block diagram for an arbitrary waveform generation and following signal processing (e.g. modulation) functions.

The *acoustic* benchmark is not limited by frequency (kHz, very easy for 350nm CMOS and scaled down technologies), but measures enabling parallel, real-time ML resources at this operating frequency. The front-end computation of 16 to 32 bands results in 16 to 32 CABs in the SoC FPAA with further optimizations expected in next-generation FPAA designs. The classifier (e.g., VMM+WTA) support in the SoC FPAA is 1 CAB per output symbol, although the next designs based on analog standard cells would routinely expect 8 to 16 symbol outputs per CAB. The issue is building an ML classifier (40nm, 5mm × 5mm) with 2.5M classifier parameters (e.g. Weights) running the very low-power (μW) 2.5GMAC(/s) algorithm (Fig. 11).

The *filter* benchmark is not limited by the number of CAB components, where this design is roughly 10 CABs of Transconductance Amplifiers (TA) devices. This benchmark is limited by the fabric frequency, the quality of

the circuits (linear range, power supply rejection), and the precision setting parameters. The SoC FPAA (350nm) has sufficient programming precision for these metrics. Case I (300kHz) and potentially Case II (3MHz) are possible in an SoC FPAA. Scaled CMOS nodes (Fig. 11) should handle Case III (130nm CMOS) and Case IV (40nm CMOS).

The *imager* benchmark is limited by the CMOS imager choice and the number of parallel nodes for image computation. Local motion processing using 10,000 CABs enables a 16×16 image block within each CAB (3MPixel imager), and in turn, enables a wide range of video/motion processing. With smaller networks, motion issues will be constrained by full-image storage. Case I and Case II focus on single image processing, while Case III and Case IV shift to motion/video processing corresponding to FPAA structures at 10,000 CAB level (Fig. 11).

3.6 Additional Potential Benchmarks

This initial set of benchmarks can be expanded in the future to evaluate additional analog & mixed-signal computational aspects and likely will grow as computational algorithms grow (Fig. 7). A potential set of future benchmark problems are briefly discussed in the following next paragraphs.

Generalized Linear and Nonlinear Dynamics. Analog computation enables a range of linear and nonlinear dynamics. The solution to $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ problem through a differential equation $\tau \frac{d\mathbf{x}}{dt}$ enables the construction of linear control systems as well as certain potential nonlinear dynamical systems. These nonlinear dynamics could include optimization networks (e.g. Hopfield) or coupled chaotic systems. The ODEs in analog computation can be extended to PDEs that could be solved on configurable devices, including parabolic (e.g. diffusion/heat equation) or hyperbolic (e.g. wave propagation) equations. These techniques show significant numerical advantages for analog computation, as digital approaches require solving $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ with a quadratically increasing condition number [70].

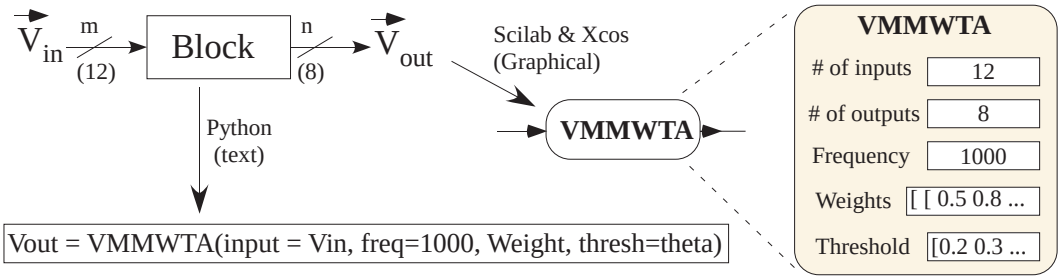


Fig. 13. An algorithm block is described by its computational function (physical devices), internal programmed parameters (e.g. FG values), and the signal busses of vectorized block inputs (e.g. $\mathbf{V}_{in}$) and outputs (e.g. $\mathbf{V}_{out}$). This block description are formulated in a graphics-based formulation (e.g. Xcos & Scilab) where the parameters are available graphically for each block, or are formulated in a text-based formulation that includes the parameters (e.g. Python routine syntax). Both formulations enable straightforward methods for macromodel simulations.

Neuron Modeling and Neuromorphic Computation. These directions are an important emerging application space (e.g. [90]) which are beyond discussions of VMMs, general ODE solutions, and solutions of linear equations, but rather are focused on the further improvement of neural algorithms toward engineering problems *beyond* the capability of analog computation [90]. Traditional easy models of rate encoded signals and integrate and fire neurons are not the desired model in these areas (e.g. [90]). The uncertain nature of these emerging computations makes defining benchmarks in this area premature, although certainly they will be arising with continued research.

Signal Generation and Computation. Many computations (analog or digital) require the generation of parallel arbitrary signals that are multiplied with one or multiple signals (Fig. 12). These techniques have multiple communication applications (e.g. OFDM, synchronized matched filters, security keys) that require arbitrary

waveform generators with modulators and filters. Analog approaches can utilize dense memory elements where each stored analog value can be played in sequence (e.g. [31, 91, 92]).

4 ANALOG ALGORITHMS ENABLING ANALOG BENCHMARKS

Between the Benchmarks and the configurable FPAA devices is the analog algorithm representation. The benchmark problems set some of the user-facing algorithm blocks that aggregate to benchmark solutions, enabling a top-down design methodology for analog computation similar to digital computational flows. These algorithm blocks make analog system design accessible to digital algorithm designers. A designer only needs to understand a few additional concepts but not the whole field of analog circuit design. These abstractable blocks, similar to digital computation, are effectively considered recognizable computations. These algorithm definitions provide a set of user-facing algorithms abstracting analog computation from most users. A coding framework enables some level of benchmark portability, although the number of options for implementation is small at the writing of this paper (e.g. FPAAs, early synthesis); we expect that further tool development can accelerate these opportunities and widen the use of these concepts. These algorithmic functions transform the expected lived experience of analog design.

An algorithm block can be represented either as a text representation (e.g. a Python function call) or as a graphical representation (e.g. Xcos) where the block parameters can be selected through the graphical interface (Fig. 13). All of the algorithm blocks are vectorized (buses are abstracted, Fig. 6) level=1 (Fig. 4) blocks with the expected vectorization that can have a Python-level definition. Graphical blocks illustrate the computation dataflow (level=1), although the text-based formulation (Python functions) is functionally equivalent (Fig. 13)). Each block should have a macromodel for potential digital simulation. Traditional circuit design for developing new algorithm blocks (level=1) is enabled through separate level=2 current-voltage representations and simulation, where eventually a resulting level=1 block is defined and macromodelled. Analog circuit design becomes complete when a circuit design is abstracted into a tool flow.

The Benchmark application is instantiated into a sequence of algorithm-level blocks specified either as a graphical or text framework. Graphical compilation from Xcos/Scilab graphical representation (Fig. 13) to targeted FPAA code [57, 61, 93] is a process utilized by multiple research groups. Recent efforts have developed a generalized synthesis tool flow (Fig. 14) from Python to Verilog to Netlist to a switch list (targetable FPAA file) through an FPAA description file as well as to IC layout (.gds) through programmable analog standard cells [62, 63] [3]. Algorithm blocks enable the infrastructure for multiple levels of architectural exploration for developing new configurable systems (e.g., for digital: OpenFPGA [94]) enabling high-level (algorithm simulation) and lower-level (device simulation) comparisons.

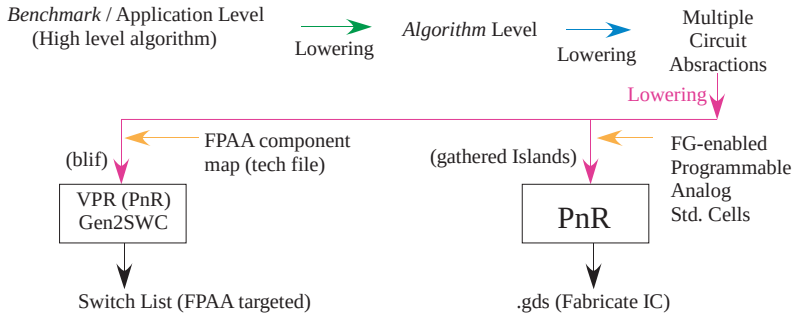
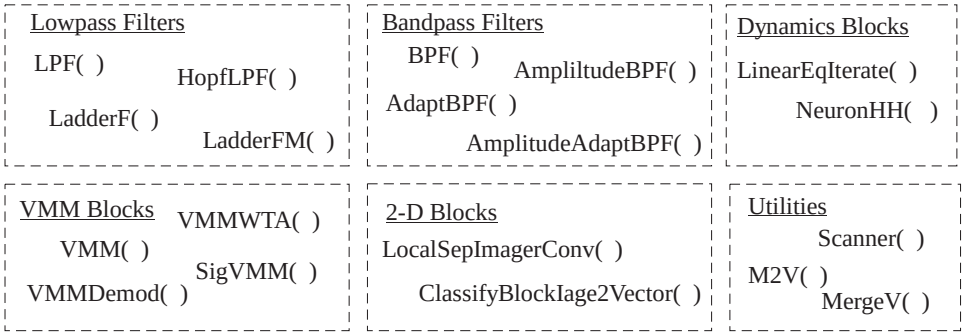


Fig. 14. Benchmark or Application level and Algorithm level within a high-level perspective of an programmable analog synthesis tool flow either to target an FPAA or to create the design for a new IC.

[3] the synthesis flow to an FPAA or to custom layout from these Python definitions are currently under review and will be made available when they are accepted



(a)

Function Area	Function	Input	Output	System Parameters	Circuit Parameters
Lowpass Filters:	LPF	n	n	cornerFreq(n)	Ibias(n)
	HopfLPF	1	n	centerFreq(n), Q(n)	Ibias1(n), Ibias2(n)
	LadderF	1	n	delay(n)	Ibias1(n), Ibias2(n)
	LadderFM	m	n×m	delay(n)	Ibias1(n), Ibias2(n)
Bandpass: **Filters**:	BPF	1	n	centerFreq(n), Q(n)	Ibias1(n), Ibias2(n)
	AdaptBPF	1	n	centerFreq(n), Q(n)	Ibias(n), Iq(n)
	AmplitudeBPF	1	n	centerFreq(n), Q(n), AmpDetect(n)	Ibias1(n), Ibias2(n), IbiasA(n)
	AmplitudeAdaptBPF	1	n	centerFreq(n), Q(n), AmpDetect(n)	Ibias(n), Iq(n), IbiasA(n)
VMM: **blocks**	VMM	m	n	W(m×n), freq(1)	Ibias(n×m)
	VMMWTA	m	n	W(m×n), Thresh(n), freq(1)	Ibias(n×m), Ithresh(n)
	SigVMM	m	n	W(m×n), freq(1)	Ibias(n×m)
	VMMDemod	m+1	n	W(m×n), mod(1), freq1(1), freq2(1)	Ibias(n×m), Ibias2(n)
Dynamics Blocks	LinearEqIterate	n	n	A(n×n), b(n), freq(1)	IbiasA(n×n), IbiasB(n)
	NeuronHH	m	n	W(m×n)	Isyn(m×n)
2-D Blocks	LocalSepImagerConv	1	m	vert=V1, horiz=h1,clock	Ivert, Ihoriz, clock
	ClassifyBlockImage2Vector	m	n	clock,	clock
Utilities	Scanner	m+2	1	clock(1), data(1)	
	M2V	m×n	m n		
	MergeV	m , n	m+n		

(b)

Fig. 15. A partial list of algorithm functions for implementing benchmarks. (a) Several functions gather into multiple function areas. (b) Greater detail on the algorithm (.py) level=1 function definitions. Each of these blocks would have a parallel definition for the highest level in Verilog-AMS with the same names. In addition to these *signal* values, there are a number of parameter values for each function.

The range of level=1 algorithm computation (Fig. 15) includes programmable first- and second-order LPF & BPFs, a range of VMM-enabled computations, nonlinear dynamics, and utility functions. These blocks have been implemented often on existing FPAAs (Fig. 18), including in the SoC FPAA. Each function has system parameters and circuit parameters, where the system parameters can be lowered to the circuit parameters. The LPF includes HopfLPF second-order block with programmable nonlinear dynamics and Ladder filter (LadderF, LadderFM) stages used for filters, delay stages, as well as computation for this effective 1-D space, 1-D time second-order PDE. The BPF includes adaptive BPF elements (AdaptBPF) as well as elements that perform a

Sample Acoustic Benchmark Code

```
y1 = AmplitudeBPF(In, center=f1, Q=Q1, fpeak = f2 )
y2 = M2V( LadderFM(input=y1,Ibias1=Ib1,Ibias2=Ib2) )
Out = VMMWTA(input=y2,W=W1,Thresh=theta1,freq=400)
```

Sample Vision Benchmark Code

```
y1 = LocalSepImagerConv(input=In,vert=V1,horiz=h1,clock=CK1)
y2 = ClassifyBlockImage2Vector(input=y1, ,clock=CK1)
Out = VMMWTA(input=y2,W=Wimage,Thresh=theta2,freq=60)
```

Sample Analog Benchmark Code

```
y1 = LadderF(input=In,Ibias1=If1,Ibias2=If2)
Out = y1(n)
```

Sample Communication Benchmark Code

```
y1 = LadderF(input=In,Ibias1=If1,Ibias2=If2)
y2 = VMMDemod(input=y1,W=W1, mod=Signal1, freq1=2.5e9, freq2=1e7)
Out = VMMWTA(input=y2,W=W1,Thresh=theta3,freq=1e7)
```

Fig. 16. Example Code Blocks for four benchmarks using analog algorithms, including a sample Acoustic, Vision, Communication, and Analog Benchmark Code. These blocks include the parameters as well as the computational flow.

tunable amplitude detection after the BPF (AmplitudeBPF, AmplitudeAdaptBPF). A BPF element will have system parameters (parameter vectors centerFreq, Q) that directly set programmable circuit biases (parameter vectors Ibias1, Ibias2) for Transconductance Amplifiers (TA) through other circuit parameters (e.g., load capacitances). As a VMM operation typically has voltage-in and current-out of the crossbar, such a computation is not a level=1 operation [45, 59], requiring either a merge with the next level of computation (e.g. WTA: VMMWTA, sigmoid for NN: SigVMM) or a current-to-voltage (e.g. a transimpedance amplifier) to create voltage outputs. The *freq* parameter specifies the maximum operating frequency and enables tools to convert the system parameters (W) to circuit parameters (Ibias) through the circuit components (e.g. capacitance). A range of linear and nonlinear dynamics are available through more advanced blocks, such as the linear equation solution (LinearEqIterate) that also enables creating a linear system definition for control systems, the Hodgkin-Huxley (HH) Neuron and synapses (NeuronHH), or using the SigVMM block to build Hopfield recurrent dynamics. Several computation utilities (e.g. M2V = Matrix to Vector transformation) enable reformatting the data where required. The blocks are not an exhaustive list but highly representative of currently defined functions (Fig. 15a); additional algorithm blocks are expected moving forward. The physical implementation involves a range of CAB/CLB utilization depending on the particular FPAA device (e.g. SoC FPAA, Fig. 18) as well as various areas depending on the programmable analog standard cell library available (e.g. [63]).

The algorithm blocks can fully specify solutions for the four benchmark problems. The *vision benchmark* primarily uses the imager convolution algorithm, ImageVMM( ), which performs a separable image transform. Commercial imagers communicate a single sample at a time by scanning the entire image sensor array. The imaging architecture should utilize this data format as much as possible to minimize the communication cost (e.g. [33, 74]). The first stage of these algorithms is a set of 2-D separable transforms with a WTA block enabling the first layer of classification. The resulting layers of transforms, classification, and compression reduce the image into a vector metric, enabling a VMM+WTA classifier for the measured result. For the communication benchmark for a single input (all are defined for multiple inputs, so using LadderFM instead of LadderF), y1 is a vector (e.g. qw) corresponding to Ibias1 and Ibias2 creating delays along the line. These spatio-temporal beamforming (VMM) are input into a parallel demodulation system (e.g. 32), and are passed to a classifier to generate multiple classified output signal lines in a vector (e.g. 64). This system also requires a demodulation signal (Signal1) that is an input modulation signal in the 2 to 2.5GHz range. One can specify that all of the four benchmark cases (defined) have only a single analog input (In). The imager input could be analog or digital, but given it is one signal, a single DAC can handle the conversion, so we will consider all of the questions with a single scalar signal input (In). The analog benchmark would have multiple of these parallel filters.

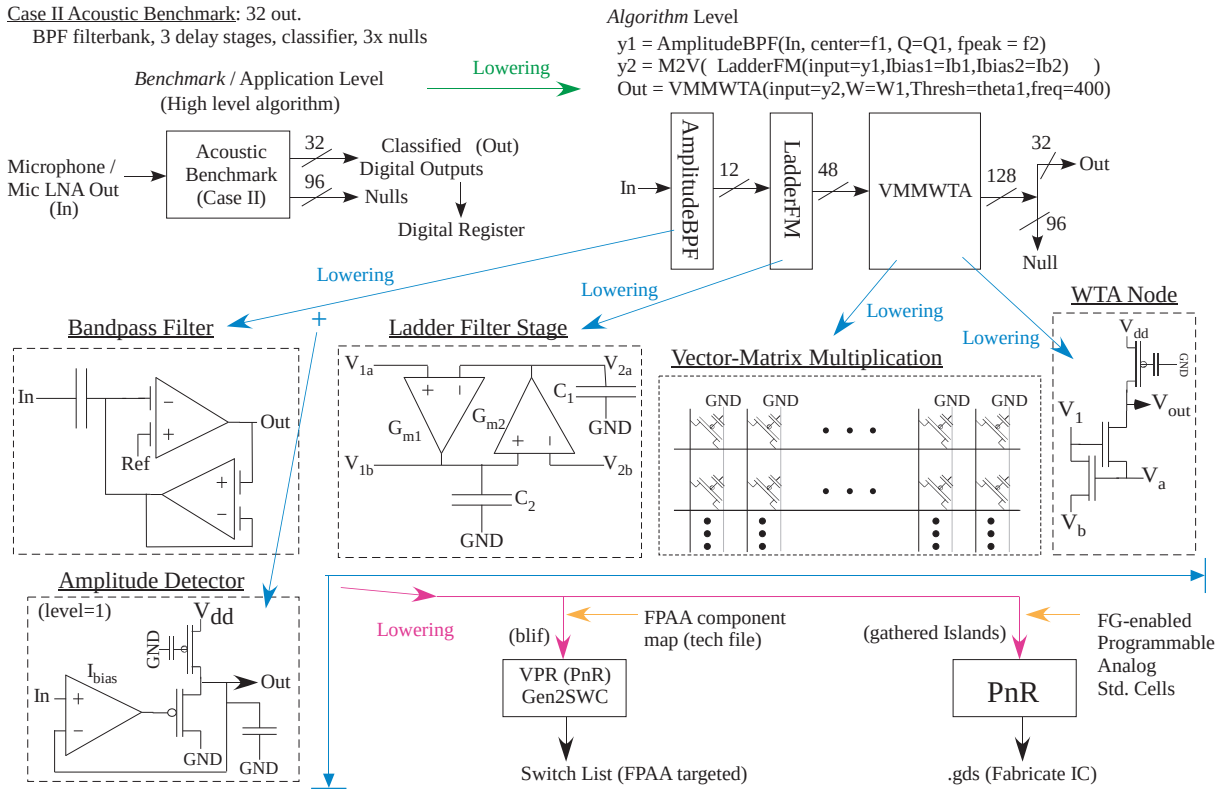


Fig. 17. Benchmark level lowering to Algorithmic Level example for an acoustic Case II implementation, classifying 32 command words from a microphone signal. This implementation uses 96 nulls, classified outputs that are not the desired 32 command words, for increased system robustness, typical of other speech recognition systems. Another user could choose to solve this benchmark application through other architectures. Most commercially available microphones already contain a Low-Noise Amplifier (LNA), therefore, the input sensor signal would come after this stage. This benchmark solution is composed of multiple algorithm-level functions, with the resulting intermediate scalar (freq), vector (Ibias1, Ibias2, theta), and matrix (W) parameters required for each stage. The flow illustrates the tool synthesis to either target an FPAA or fabricate a new IC, including lowering the AmpitudeBPF block into its component (level=1) Bandpass Filter and Amplitude Detector block circuits, lowering the LadderFM block into its component Ladder Filter Stages, and lowering the VMMWTA block into the crossbar Vector-Matrix Multiplication (VMM), often implemented in routing fabric, and WTA node circuits.

A high-level implementation example of one benchmark, the acoustic Case II benchmark, illustrates the steps starting from the benchmark instantiate in the algorithm code, and the lowering & compilation strategy (Fig. 17). The Python description gets lowered into Verilog AMS and through the resulting tools to a switch list or a .gds output. For the Case II acoustic benchmark (Fig. 17), for single microphone input, y1 is an output vector corresponding to the output feedback vector (12 lines), y2 is an output vector corresponding to three approximate delays at each BPF stage set through Ibias1 (e.g. 3 by 12) and Ibias2 (e.g. 3 by 12) creating delays (48 lines), and a 48-input by 128-output VMM+WTA classifier stage with a 6144 weight matrix. This system would require more WTA stages than would fit on an SoC FPAA (barely), although such a design could theoretically be implemented on two SoC FPAAs (350nm CMOS), and potentially easily implemented in a similar size 130nm FPAA. Using 130nm standard cells, this implementation would roughly require an area of $250\mu m \times 300\mu m$. Functions can have parameters at a higher function level, such as frequency or weights, or lowered values such as bias currents. Our synthesis tools lower functionality from higher level to lower level parameters. This example explicitly shows both cases.

Function Area	Function	Size	Previous Measure
Lowpass	LPF	1/4 CAB (TA)	[31]
Filters:	HopfLPF	1 CAB (4 TA)	[86]
	LadderF	1 CAB / stage (2FG TA)	[75]
	LadderFM		[75]
Bandpass:	BPF	1/2 CAB (2 TA)	[31]
Filters:	AdaptBPF	3/4 CAB (3 TA)	[95]
	AmplitudeBPF	1 CAB (4 TA)	[31, 96]
	AmplitudeAdaptBPF	1 CAB (4 TA)	(internal)
VMM:	VMM	**R** + 4TA (13in + 2 out/CAB)	[97]
blocks	VMMWTA	**R** + 2nFETs (13in + 1 out/CAB)	[73, 80, 98]
	SigVMM	**R** + 4 tA (13in + 4 out/CAB)	[99]
	VMMDemod		(internal)
Dynamics	LinearEqIterate	4 Matrix elements / CAB	[100]
Blocks	NeuronHH	**R** + CAB (13in + 1 out/CAB)	[101]
2-D Blocks	LocalSepImagerConv		[33]
	ClassifyBlockImage2Vector		[33, 74]
Utilities	Scanner	1 16bit shift reg / CAB	[92]
	M2V	m×n	(typical)
	MergeV		(typical)

Fig. 18. FPAA sizes for the partial list of algorithm (.py) function definitions at level=1. The FPAA sizes are baselined to the CABs in the SoC FPAA [31]. Previous CAB components were designed by intuition as most system infrastructure was not available. New CABs and new FPAA architectures and designs will be optimized utilizing benchmarks. Many functions utilize the CAB Routing (**R** for the computation. Some internal functions have not yet been published at this stage.

5 ANALOG & MIXED-SIGNAL SUMMARY AND DISCUSSION

This first analog and mixed-signal benchmark suite of various sizes and domains enables performance evaluation and optimization with recent analog & mixed-signal design & synthesis tools. These benchmarks comprise multiple levels of an **acoustic** system, a **vision** system, a **communications** system, and a classical analog **filter** system that must be simultaneously satisfied for a complete system. ML is essential in two systems, and can be utilized in the other two systems. Defining analog computation system benchmark and its underlying computation enables future automated architectural design space exploration (DSE), to determine the best configurable architecture (e.g., a new FPAA) for a certain family of applications. The demonstration of the benchmarks will further show the 1000× improvement (and more) that Mead originally discussed (1990) [46] and illustrate further ways to utilize these concepts.

Benchmarks are an indicator of the perceived fundamental computations of that particular computing substrate. Although digital computing benchmarks mostly follow the solution of linear equations, analog benchmarks utilize ODE, PDE, and some linear equation solutions. These efforts enable the evaluation of targeted FPAAs, custom ICs, and new FPAA designs through recent analog computing theory. Abstracting the capabilities of analog computing shows the resulting analog benchmarks. These benchmarks are composed of an initial set of analog & mixed-signal *algorithm* blocks that provide solutions for these particular benchmarks.

One expects analog synthesis tools to eventually parallel digital and coding synthesis tools, including enabling adding new functionalities to the libraries at different levels (Fig. 19). The lower-level tools exist to handle functions or features that are not currently in the algorithm library (Fig. 19). Tools enable building blocks with the

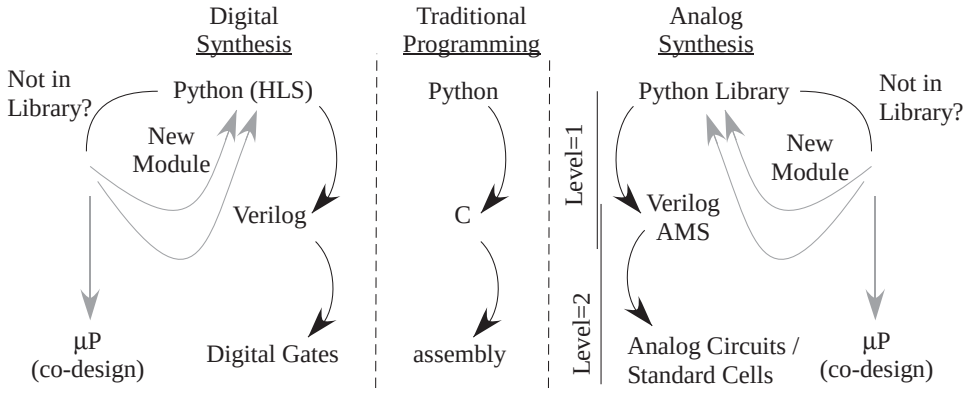


Fig. 19. Similar to traditional programming which has a hierarchy and compilation flow from higher-level languages (e.g. Python) to more machine-specific languages (e.g. C and assembly), digital and analog synthesis for configurable structures has similar flows. If a component is not in the library so that the tools cannot directly compile a block, it opens an opportunity for the design of a new module in a lower tool flow and/or direct compilation of that block into a processor co-designed with the implementation.

full analog circuit capability (e.g., level=2, [60]) that build up to new level=1 blocks as well as enabling expert designers to utilize more device-level features. A digital programmer (Fig. 19) might utilize a similar flow utilizing C code or even assembly code in a larger Python project, or an FPGA designer might utilize Verilog or gate-level design in a C++ or Python-defined project. If an exp(·) or sin(·) function is not available for an FPGA or embedded design, it could be built at a lower level and incorporated into the function library. The capability is similar to analog computation and synthesis.

Benchmarks provide new FPAA devices metrics (e.g. scaled FPAA technologies), enabling optimizing synthesis procedures through analog-digital co-design space exploration (Fig. 20). An automated architectural DSE can determine the best architecture (e.g., a new FPAA) compared to these benchmarks, adapting the compiled FPAA technology files for scaled FPAA devices (Fig. 20). The performance evaluation includes area & resource allocation, energy & power, SNR, and delay & latency. Some metrics can be directly evaluated by the synthesis procedure, while others require a level of circuit and system simulation to verify the synthesized performance. Benchmarking evaluates the proposed automation tool as well as the generated analog/mixed-signal circuit or device (Fig. 20).

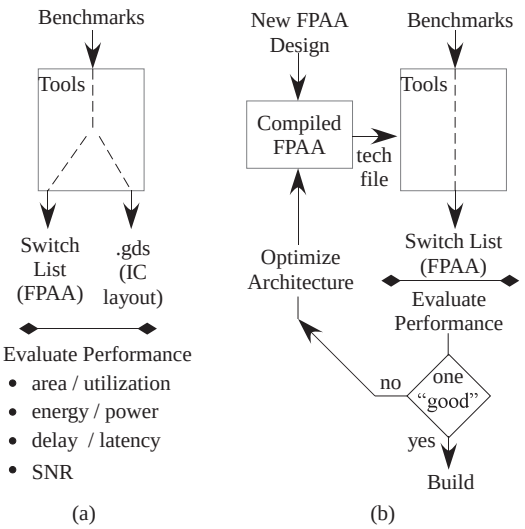


Fig. 20. Potential benchmark applications for application synthesis. (a) Benchmarks enable the evaluation of analog tool synthesis to either an FPAA or a new analog IC. (b) Benchmarks enable the design of new FPAAs by enabling tool synthesis, evaluation, and redesign.

The mixture of potential analog computation, digital computation, and specialized digital components (e.g., a μP) creates the analog-digital hardware-software co-design problem, a problem that system performance against the benchmark suite will give some intuition for particular systems designs. When should an algorithm, or part of an algorithm, be performed on a μP, in digital fabric, and in analog fabric? For

low-power computation, numerically intense computations can shift to analog components, and the processor can address the control flow and movement of data streams throughout the computation. Some functions will be explicitly specified by the designer, whereas some functions will have characterized options for both digital and analog options. Some tool flows can be easily compiled from known routines, whereas some may require more complex synthesis requiring designer interaction to utilize designers' expertise in the optimization. The question of choosing a particular resource for a function will require building a performance estimation model, including the algorithmic and architecture complexity of the analog & digital components, where the performance of various options is evaluated against the defined benchmarks.

REFERENCES

[1] J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK benchmark: Past, present and future," *Concurrent Comput.Practice Exp.*, pp. 803–820, 2003.

[2] J. Hasler, "Starting framework for analog numerical analysis for energy efficient computing," *Journal of Low Power Electronics Applications*, vol. 7, no. 17, pp. 1–22, 2017.

[3] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "CHStone: A benchmark program suite for practical c-based high-level synthesis," in *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 1192–1195.

[4] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communications systems," in *Proceedings of 30th Annual International Symposium on Microarchitecture*. IEEE, 1997, pp. 330–335.

[5] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, "Machsuite: Benchmarks for accelerator design and customized architectures," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2014, pp. 110–119.

[6] S. Verdoolaege, J. Carlos Juega, A. Cohen, J. Ignacio Gomez, C. Tenllado, and F. Catthoor, "Polyhedral parallel code generation for cuda," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 9, no. 4, pp. 1–23, 2013.

[7] Y. Zhou, U. Gupta, S. Dai, R. Zhao, N. Srivastava, H. Jin, J. Featherston, Y.-H. Lai, G. Liu, G. A. Velasquez, W. Wang, and Z. Zhang, "Rosetta: A realistic high-level synthesis benchmark suite for software programmable fpgas," in *FPGA*, feb 2018.

[8] A. Arora, A. Boutros, D. Rauch, A. Rajen, A. Borda, S. A. Damghani, S. Mehta, S. Kate, P. Patel, K. B. Kent, V. Betz, and L. K. John, "Koios: A deep learning benchmark suite for fpga architecture and cad research," in *FPL*, aug 2021.

[9] D. Stroobandt, P. Verplaetse, and J. Van Campenhout, "Generating synthetic benchmark circuits for evaluating cad tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 1011–1022, 2000.

[10] G. Ndu, J. Navaridas, and M. Luján, "Cho: towards a benchmark suite for opencl fpga accelerators," in *Proceedings of the 3rd International Workshop on OpenCL*, 2015, pp. 1–10.

[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee, 2009, pp. 44–54.

[12] K. Krommydas, W.-c. Feng, C. D. Antonopoulos, and N. Bellas, "Opendwarfs: Characterization of dwarf-based benchmarks on fixed and reconfigurable architectures," *Journal of Signal Processing Systems*, vol. 85, pp. 373–392, 2016.

[13] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd workshop on general-purpose computation on graphics processing units*, 2010, pp. 63–74.

[14] Q. Gautier, A. Althoff, P. Meng, and R. Kastner, "Spector: An opencl fpga benchmark suite," in *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2016, pp. 141–148.

[15] R. Njuguna and R. Jain, "A survey of fpga benchmarks," *Project Report, November*, vol. 24, 2008.

[16] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*. IEEE, 2001, pp. 3–14.

[17] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski, "Legup: high-level synthesis for fpga-based processor/accelerator systems," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, 2011, pp. 33–36.

[18] R. Ahmed, A. A. M. Bsoul, S. J. E. Wilton, P. Hallschmid, and R. Klukas, "High-level synthesis-based design methodology for dynamic power-gated FPGAs," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, 2014, pp. 1–4.

[19] "Vitis high-level synthesis," https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html.

[20] "Circt project under mlir framework," https://circt.llvm.org/.

[21] S. Huang, K. Wu, H. Jeong, C. Wang, D. Chen, and W.-m. Hwu, "Pylog: An algorithm-centric python-based fpga programming and synthesis flow," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2021, pp. 227–228.

[22] H. Ye, C. Hao, H. Jeong, J. Huang, and D. Chen, "Scalehls: Achieving scalable high-level synthesis through mlir," *LATTE Workshop on Languages, Tools, and Techniques for Accelerator Design*, 2021.
[23] D. W. Chang, C. D. Jenkins, P. C. Garcia, S. Z. Gilani, P. Aguilera, A. Nagarajan, M. J. Anderson, M. A. Kenny, S. M. Bauer, M. J. Schulte *et al.*, "Ercbench: An open-source benchmark suite for embedded and reconfigurable computing," in *2010 International Conference on Field Programmable Logic and Applications*. IEEE, 2010, pp. 408–413.
[24] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*, no. CONF, 2015.
[25] V. Betz, "The fpga place-and-route challenge," http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html.
[26] Wolf, "Hardware-software co-design of embedded systems," *Proceedings of the IEEE*, pp. 967–989, 1994.
[27] Q. Zhao, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "An automatic FPGA design and implementation framework," in *IEEE DAC*, 2013.
[28] M. Weinhardt, A. Krieger, and T. Kinder, "A framework for PC applications with portable and scalable FPGA accelerators," in *IEEE DAC*, 2013.
[29] D. Rossi, C. Mucci, M. Pizzotti, L. Perugini, R. Canegallo, and R. Guerrieri, "Multicore signal processing platform with heterogeneous configurable hardware accelerators," *IEEE Trans. VLSI*, vol. 22, no. 9, pp. 1990–2003, 2014.
[30] J. Hasler, "Large-scale field programmable analog arrays," *IEEE Proceedings*, vol. 108, no. 8, pp. 1283–1302, 2020.
[31] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, W. R, S. Nease, and S. Ramakrishnan, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Transactions on VLSI*, vol. 24, no. 6, pp. 2253–2261, 2016.
[32] J. Hasler, S. Kim, and A. Natarajan, "Enabling energy-efficient physical computing through analog abstraction and IP reuse," *Journal of Low Power Electronics Applications*, vol. 8, no. 4, pp. 1–23, 2018.
[33] J. Hasler, "Analog architecture and complexity theory to empowering ultra-low power configurable analog and mixed mode SoC systems," *Journal of Low Power Electronics Applications*, vol. 9, no. 1, pp. 1–38, 2019.
[34] J. Hasler and E. Black, "Physical computing: Unifying real number computation to enable energy efficient computing," *Journal of Low-Power Electronics Applications*, pp. 1–21, 2021.
[35] "OpenMSP430 project: open core MSP430," http://opencores.org/projectopenmsp430.
[36] J. Hasler, "The potential of SoC FPAAs for emerging ultra-low-power machine learning," *Journal of Low Power Electronics and Applications*, vol. 12, no. 2, 2022.
[37] Hasler, B. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate device," in *Advanced Research in VLSI*, Mar 1999, pp. 215–229.
[38] M. Kucic, Hasler, J. Dugger, and D. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Advanced Research in VLSI*, Mar 2001, pp. 148–162.
[39] T. Hall, D. Anderson, and Hasler, "Field-programmable analog arrays: A Floating-Gate approach," in *Field-Programmable Logic (FPL) and Applications*. Montpellier, France: Springer-Verlag, 2002, pp. 424–433.
[40] M. A. Brooke, "A reconfigurable general purpose analog integrated circuit," Ph.D. dissertation, University Southern California, 1988.
[41] M. A. Sivilotti, *Wiring Considerations in Analog VLSI Systems, With Application to Field-Programmable Networks (VLSI)*. Ph.D., California Institute of Technology, Pasadena, CA, 1991.
[42] E. K. F. Lee and P. G. Gulak, "A CMOS field programmable analog array," *IEEE JSSC*, pp. 1860–1867, 1991.
[43] H. W. Klein, "The EPAC architecture: an expert cell approach to field programmable analog circuits," in *IEEE Midwest CAS*, 1992, pp. 169–172.
[44] Anadigm, "Specifically generic analog functions for fpaas: Anadigm says," *EE Times*, 2004.
[45] C. Schlottmann and Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal on Emerging CAS*, vol. 1, no. 3, pp. 403–411, 2011.
[46] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, no. 78, pp. 1629–1636, 1990.
[47] V. Srinivasan, G. Serrano, C. Twigg, and Hasler, "Floating-gate-based programmable CMOS reference," *IEEE Transactions CAS I*, vol. 55, no. 11, pp. 3448 – 3456, 2008.
[48] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Models and techniques for temperature robust systems on a reconfigurable platform," *Journal of Low Power Electronics Applications*, vol. 7, no. 21, pp. 1–14, 2017.
[49] S. Peng, G. Gurun, C. Twigg, M. Qureshi, A. Basu, S. Brink, Hasler, and D. F.L, "A large-scale reconfigurable smart sensory chip," in *IEEE ISCAS*, 2009, pp. 2145 – 2148.
[50] M. Laiho, J. Hasler, J. Zhou, C. Du, W. Lu, E. Lehtonen, and J. H. Poikonen, "FPAA/Memristor hybrid computing infrastructure," *IEEE Transactions CAS I*, 2014.
[51] C. Twigg, J. Gray, and Hasler, "Programmable floating-gate FPAA switches are not dead weight," in *IEEE ISCAS*, 2007, pp. 169–72.
[52] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation,," *IEEE JSSC*, vol. 42, no. 2, pp. 280–291, 2007.
[53] J. Hasler and H. Wang, "A fine-grain FPAA fabric for RF + Baseband," in *GOMAC*, 2015.

[54] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ics," *IEEE Transactions VLSI*, vol. 24, no. 6, pp. 2244–2252, 2016.
[55] J. Hasler and S. Shah, "Security implications for ultra-low power configurable analog and mixed mode SoC systems," *Journal of Low Power Electronics and Applications*, pp. 1–17, 2018.
[56] J. Hasler, S. Kim, and F. Adil, "Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems," *Journal of Low Power Electronics Applications*, July 2016.
[57] M. Collins, J. . Hasler, and S. George, "An open-source toolset enabling analog–digital software codesign," *Journal of Low Power Electronics Applications*, vol. 6, no. 1, pp. 1–15, 2016.
[58] C. R. Schlottmann, C. Petre, and P. E. Hasler, "Simulink framework for design to and automated conversion on large-scale FPAA devices," *IEEE Transactions VLSI*, 2011.
[59] C. Schlottmann and J. Hasler, "High-level modeling of analog computational elements for signal processing applications," *IEEE Transactions VLSI*, vol. 22, no. 9, pp. 1945–1953, 2014.
[60] A. Natarajan and J. Hasler, "Modeling, simulation and implementation of circuit elements in an open-source tool set on the FPAA," *Analog Integrated Circiuits and Signal Processing*, vol. 91, no. 1, pp. 119–130, 2017.
[61] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz, "Vtr 7.0: Next generation architecture and CAD system for FPGAs," in *IEEE ASIC Seminar*, vol. 7, 2014, pp. 6:1–6:30.
[62] J. Hasler, "Defining analog standard cell libraries for mixed-signal computing enabled through educational directions," in *IEEE ISCAS*, 2021.
[63] J. Hasler, B. Muldrey, and P. Hardy, "A CMOS programmable analog standard cell library in skywater 130nm open-source process," in *WOSET*, Oct 2021.
[64] C. Mead and L. Conway, *Introduction to VLSI System Design*. Addison-Wesley, 1980, early drafts found at http://ai.eecs.umich.edu/people/conway/VLSI/VLSIText/VLSIText.html.
[65] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, 1965.
[66] ——, "Progress in digital integrated electronics," *IEEE IEDM*, pp. 11–13, 1975.
[67] B. Hoeneisen and C. A. Mead, "Fundamental limitations in microelectronics – i. mos technology," *Solid State Electronics*, vol. 15, no. 7, pp. 819–829, 1972.
[68] R. Turing, "On computable numbers," *Proceedings of the London Mathematical Society,*, pp. 230–265, 1937.
[69] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128x32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *CICC*, 2004, p. 651.
[70] J. Hasler and A. Natarajan, "Continuous-time, configurable analog linear system solutions with transconductance amplifiers," *IEEE Circuits and Systems I*, vol. 68, no. 2, pp. 765–775, 2021.
[71] M. Demler, "Mythic multiplies in a flash: Analog in-memory computing eliminates dram read/write cycles," *The Linley Group*, August 2018.
[72] S. Brink, J. Hasler, and R. Wunderlich, "Adaptive floating-gate circuit enabled large-scale FPAA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2307–2315, 2014.
[73] J. Hasler and S. Shah, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE Journal on Emerging CAS*, vol. 8, no. 1, pp. 28–37, 2018.
[74] S. Koziol, "Reconfigurable analog circuits for autonomous vehicles," Ph.D. dissertation, Georgia Institute of Technology, 2013.
[75] J. Hasler and S. Shah, "An SoC FPAA based programmable, ladder-filter based, linear-phase analog filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 592–602, 2021.
[76] J. S. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren, "Timit acoustic-phonetic continuous speech corpus," in *Linguistic Data Consortium, Philadelphia, Available from https://github.com/philipperemy/timit*, 1983.
[77] R. G. Leonard and G. R. Doddington, "Ti digits database," https://catalog.ldc.upenn.edu/docs/LDC93S10/tidigits.readme.html, 1993.
[78] P. Warden, "Speech commands: A public dataset for single-word speech recognition." *arXiv: arXiv:1804.03209. Available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz*, 2017.
[79] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.
[80] S. Shah and J. Hasler, "VMM + WTA embedded classifiers learning algorithm implementable on SoC FPAA devices," *IEEE Journal on Emerging CAS*, vol. 8, no. 1, pp. 65–76, 2018.
[81] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: A case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, vol. 2, 1994, pp. 77–82 vol.2.
[82] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 740–755.

[83] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
[84] Q.-Y. Jiang, Y. He, G. Li, L. L. Jian Lin, and W.-J. Li, "SVD: A large-scale short video dataset for near-duplicate video retrieval," in *Proceedings of International Conference on Computer Vision*, 2019.
[85] R. Rajashekar, M. Di Renzo, K. Hari, and L. Hanzo, "A beamforming-aided full-diversity scheme for low-altitude air-to-ground communication systems operating with limited feedback," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6602–6613, 2018.
[86] J. Hasler, "A programmable on-chip hopf bifurcation circuit," *IEEE CAS I, on IEEE Xplore*, 2022.
[87] R. L. Geiger and E. Sánchez-Sinencio, "Active filter design using operational transconductance amplifiers: A tutorial," *IEEE Circuits and Devices Magazine*, vol. 1, pp. 20–32, 1985.
[88] P. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, 2nd ed. Oxford University Press, 2002.
[89] C. Cuypers, Y. Voo, M. Teplechuk, and J. Sewell, "General synthesis of complex analogue filters," *Circuits, Devices and Systems, IEE Proceedings -*, vol. 152, pp. 7 – 15, 03 2005.
[90] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in Neuroscience*, vol. 7, 2013.
[91] C. Schlottmann, S. Shapero, S. Nease, and Hasler, "A digitally-enhanced reconfigurable analog platform for low-power signal processing," *IEEE JSSC*, vol. 47, no. 10, pp. 2174–2184, 2012.
[92] J. Hasler and S. Shah, "Security implications for ultra-low power configurable analog and mixed mode SoC systems," *Journal of Low Power Electronics and Applications*, pp. 1–17, 2018.
[93] S. Kim, S. Shah, R. Wunderlich, and J. Hasler, "CAD synthesis tools for floating-gate SoC FPAAs," *Design Automation for Embedded Systems*, pp. 1–16, 2021.
[94] X. Tang, E. Giacomin, A. Alacchi, B. Chauviere, and P.-E. Gaillardon, "Openfpga: An opensource framework enabling rapid prototyping of customizable fpgas," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 367–374.
[95] J. Hasler, "A programmable adaptive-q bpf circuit," *second revision to IEEE CAS I*, 2023.
[96] S. Shah and J. Hasler, "Tuning of multiple parameters with a BIST system," *Journal of Low Power Electronics Applications*, vol. 64, no. 7, pp. 1772–1780, 2017.
[97] A. Nagarajan and J. Hasler, "Built-in self-test of vector matrix multipliers on a reconfigurable device," in *IEEE ISCAS*, may 2020.
[98] S. Shah and J. Hasler, "Low power speech detector on a FPAA," in *IEEE ISCAS*, 2017.
[99] P. Mathews and J. Hasler, "Physical computing for hopfield networks on a reconfigurable analog ic," in *IEEE ISCAS*, may 2023.
[100] J. Hasler and A. Natarajan, "Continuous-time, configurable analog linear system solutions with transconductance amplifiers," *IEEE Circuits and Systems I*, vol. 68, no. 2, pp. 765–775, 2021.
[101] A. Natarajan and J. Hasler, "Implementation of synapses with Hodgkin-Huxley neurons on the FPAA," in *IEEE ISCAS*, may 2019.