FPAA Empowering Cooperative Analog-Digital Signal Processing

Craig Schlottmann and Paul Hasler

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia
Email: cschlott@gatech.edu, phasler@ece.gatech.edu

***Abstract*—Large-scale field programmable analog array (FPAA) ICs have made analog and analog-digital signal processing techniques accessible to a much wider community. Given this opportunity, we present in this paper a framework for considering analog signal processing techniques for low-power, portable systems. These techniques have become more important given the recent recognition of the power-efficiency wall for commercial digital ICs. The discussion focuses on the framework technique required to enable analog-digital signal processing techniques. A framework is needed to enable system designers to directly develop applications into these approaches that includes considering power consumed, system complexity/area, as well as other commercial metrics. A key part of this discussion is evolving existing Simulink FPAA design tools to work with this framework such that users have a similar experience one expects with digital system design, as well as model closely experimental data at a high-level framework. The result of these techniques is pulling analog computation towards the system level development as seen in digital system design over the last 30 years.**

I. Analog Signal Processing with FPAAs

Cooperative analog-digital signal processing (CADSP) is the design approach whereby the two domains (analog and digital) are used in combination to achieve advanced system performance [1]. In traditional systems, analog processing (ASP) is used primarily for front-end amplification and data conversion, whereas digital processing (DSP) handles the mathematical operations. By repartitioning the boundary between the processing domains, we stand to take advantage of extreme power and area savings. For instance, the natural physics of the subthreshold transistor can be used to perform many mathematical operations with a fraction of the number of devices required for digital computation [2] and a much lower total current draw. In terms of MMACS/mW, we have seen analog computers achieve a 10,000x increase in power efficiency over their digital counterpart, a 20-year leap on the Gene's Law curve [3], [4].

It is the efficient balancing of the analog and digital domains that the highest performance can be achieved. A popular subset of this concept is the notion of *digitally-enhanced analog systems*, whereby digital processing is utilized to add resolution to analog blocks [5]. As a broader approach, CADSP additionally promotes the use of analog techniques to increase the power efficiency of digital blocks, as illustrated in the system of Fig 1. CADSP techniques have been successfully utilized in compressive sensing [6] and classifier systems [7]. However, two very important hurdles have prevented the wide-spread use of analog computation: the lack of programmability and the absence of robust design tools.

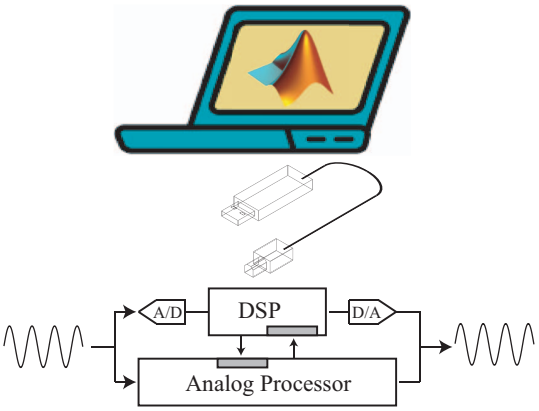

Fig. 1: The analog processor embedded with a digital processor provides a power efficient platform.

The recent development of large-scale field programmable analog arrays (FPAAs) has provided a stable platform for programmable analog systems [8]. The reconfigurable analog signal processor (RASP) FPAA is a VLSI system that contains hundreds of configurable analog blocks (CABs) and tens of thousands of programmable floating-gate transistors (FG) in a cross-bar switch matrix (SM). This flexible architecture allows the user to program the FG switches in such a way as to connect the analog components in any configuration. The RASP FPAA has demonstrated such systems as low power vector-matrix multiplier (VMM) [9] and OFDM system [10]. Trends in FPAA development are shown in Fig. 2.

The FPAA has provided the hardware platform to develop ASP systems, but the remaining fundamental problem is that it is not always easy for the typical DSP engineer to utilize analog techniques. To solve this problem, we are developing the design tools needed to empower the non-circuit designer to take advantage of the FPAA hardware. We have chosen MATLAB Simulink as the top level design space for analog systems on FPAAs [11] in order to appeal to the broadest audience of DSP engineers. Although this design space is intuitive and makes systems easy to visualize and simulate, there is still no established framework for the proper abstraction of analog design. The development of a high-level framework for abstracting analog design and creating behavioral analog blocks is necessary to bridge the analog and digital design for the system engineer.

The goal of this paper is to define such a standard analog abstraction method for the purposes of high-level system design. There are several key challenges that must be overcome in order to make analog design accessible to the system designer,

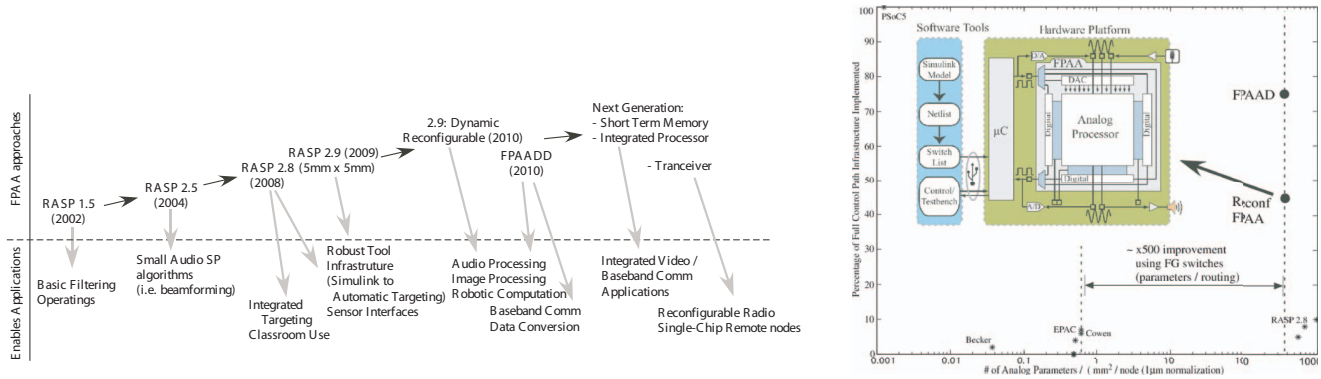

Fig. 2: High level viewpoint on Large-Scale FPAA devices. (a) Roadmap of the current and proposed GT FPAA IC development. Each generation enabled another leap in potential computation and signal processing; we expect the next generation to have yet another dramatic effect on the range of potential possible applications. Dates are when the device was first demonstrated; some devices are still in review. (b) Plot of the Percentage of Control Path implemented versus Analog Parameter Density. Recent FPAA ICs, like the dynamically reconfigurable FPAA or FPAADD device, begin to effectively maximize both parameters.

which we address in the remainder of this paper. Section II covers the technique of analog abstraction and system-level constraints. In Section III, we use a bottom-up example to show how the circuit modeling procedures are deployed to create a functional block. Lastly, we use Section IV for concluding remarks.

II. Analog Abstraction Concepts

In this section, we describe several of the high-level design choices that were made in creating the CADSP framework. From choosing a top-level design space, to constraining the interface between blocks, a well-planned framework facilitates the design of large-scale systems.

A. Simulink/High-level

Simulink is used as the top-level design space for analog signal processing in the RASP FPAA [12]. The use of Simulink was important to us because it is already a familiar tool to many DSP and control-system engineers. The intuitive nature of high-level blocks with wires in between makes it easy to design at the system-level. The Simulink tool has proven to be an intuitive interface for graphical analog design and has been used extensively in a graduate-level analog system design course [13].

Simulink comes prepackaged with many libraries of components, yet lacks high-level analog blocks. Therefore, we needed to create our own libraries for custom ASP blocks. The tool framework allows the analog engineer to easily add new blocks to the analog libraries. The key with block design is that the system should be modeled at the behavioral level, so that it is easy for the system engineer to place into a larger design. The ASP libraries promote the reuse of well-tested circuits as well as the propagation of expertise.

The creation of high-level blocks introduces the question of how much abstraction is required. If large mixed-mode systems are to be simulated, we need to provide macromodels for each analog block. Macromodels serve to reduce the simulation time and may include options as to how many second-order effects to include (such as noise and distortion). Circuit abstraction also means that we should cover up the detailed circuit parameters by fixing all of the static parameters and presenting the user-defined parameters as they relate to the system specifications.

B. Voltage mode systems

The first step in making analog design feel like digital design is to define a standard protocol for the interface between blocks. Digital design benefits from a very simple convention of high and low voltages. Conversely, analog systems can propagate information by means of small, large, voltage, or current signals. In general, these operating domains create advantages for analog systems. As illustrated in Fig. 3, current-mode can easily sum signals, while voltage-mode can broadcast signals to many destinations. Although each domain has its advantages, these choices are exactly what we want to abstract away so that things are easy and familiar to the digital designer.

At the expense of current-mode's efficient summing, we constrain the interface of our Simulink blocks to voltage-mode operation. This constraint is more like traditional digital design where a single block can fan out to many, but signals must be summed through a device, not simply shorted. We can still take full advantage of the current-mode analog processing inside the block, but the interface is exclusively voltage.

The voltage-mode design methodology has implications on the up-front design of each analog block. Many analog systems have a native current-mode interface, in which case we will embed conversion stages. The V/I or I/V stages can take many forms, and the best choice will depend on the particular application or specification. Within each block, we generally characterize multiple conversion choices so that the user can select the one they want based on the performance.

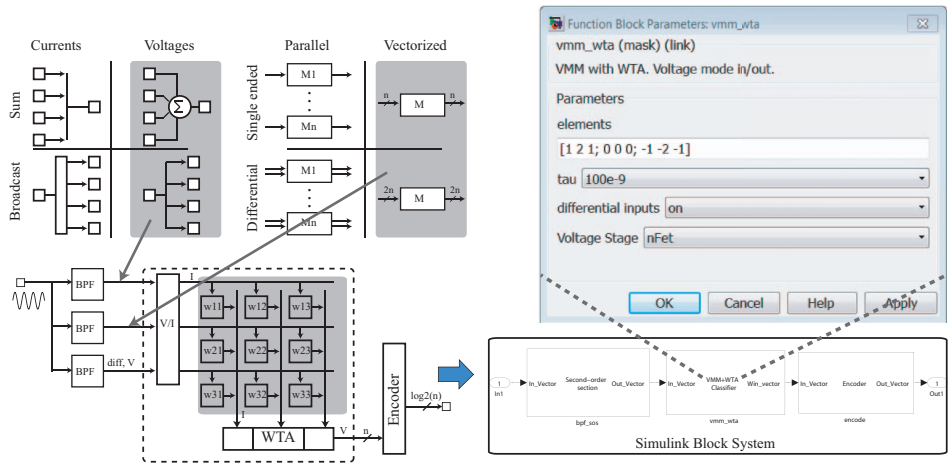

Fig. 3: The system abstraction involves defining our signal interface. We constrain the analog processing tool to use only voltage-mode lines between blocks because it is more similar to digital design and fits into the Simulink framework. Vectorized signals are also important because it takes advantage of the analog processor's parallel processing capabilities. These properties are illustrated in the classifying system at the bottom and built into high-level blocks to the right. Each block has a GUI where the functional parameters can be set without knowledge of the underlying circuits and biases.

C. Vectorized signals

Frequently in DSP, and in particular when using MATLAB, the lines between blocks are vectorized. This is common in matrix operations where the inputs are all in parallel. We have incorporated this vectorized net aspect into the analog tool structure. Although a size of one is often sufficient, each net can have any size vector dimension. Rather than forcing the user to define every size, the signals are automatically scaled based on the blocks that are used. For example, if an $M \times N$ VMM is instantiated, the input vector will automatically have a size N, and the output will have size M.

Fig. 3 illustrates the use of differential along with single-ended vectorized lines. Often in analog design, differential signals are used to increase SNR or cancel even-order harmonics. To keep the design simple, single-ended or differential mode can be selected inside a block as a parameter without changing the complexity of the blocks in the design window.

D. Biasing

A major design element of analog systems is the proper biasing of the blocks. This is a concept that is not manifest in digital design, and therefore must be dealt with behind the scenes.

The RASP line of FPAAs is built in a network of floating-gate switch elements. This is a very useful element, as it can also store bias values for computation (one of the reasons such high computational density is achieved). The analog designer can store the FG bias values inside the block, without necessitating input for the end user. Often though, the bias value is derived from a parameter in the system's function. For instance, in a G_mC filter, the time constant is given by a C/G_m relation. The G_m is set through the bias current (I_b) of the OTA such that $G_m = I_b\kappa/(2U_T)$. This simple equation can be written into the block, so that the user only needs to specify the time constant, and the correct bias will be programmed.

III. System Example: Oscillator Circuit Model

We use the oscillator design in Fig. 4 as an example to illustrate the high-level analog modeling. To begin the analog design process, the user should look to the available elements in the Simulink analog libraries. Two such libraries are shown: Level 1 and Level 2.

The Level 1 library contains the high-level system blocks. These blocks conform to the voltage-mode protocol and contain sufficient abstraction so that they are reasonable to simulate in Simulink.

The Level 2 library contains the low-level blocks, typically mapping directly to FPAA CAB elements. These blocks do not conform to the voltage-mode protocol and might have advanced modeling parameters. These blocks are best used by circuit-design engineers and should be simulated in a SPICE environment.

Additional digital libraries are not shown in the figure, but are acceptable for use in FPAA mixed-mode design. The RASP FPAA is capable of compiling these digital circuits if an accurate circuit model is attached to each block. Alternatively, if proper FPAA ports are specified, mixed-mode designs can be divided such that the entire system is simulated in Simulink and only the analog portions are compiled to the FPAA.

To add a functional block to the Level 1 library, an analog designer will start with the Level 2 blocks. A second-order-section oscillator is shown in the bottom right of Fig. 4. This system contains two FG-input OTAs, one OTA, two capacitors, two ground connects, and one FG element to short the feedback path. This last element, the FG short, demonstrates one difficulty in performing current-mode operations Simulink. The feedback in this circuit mixes two currents and integrates them on the left capacitor. Although mixing currents is a

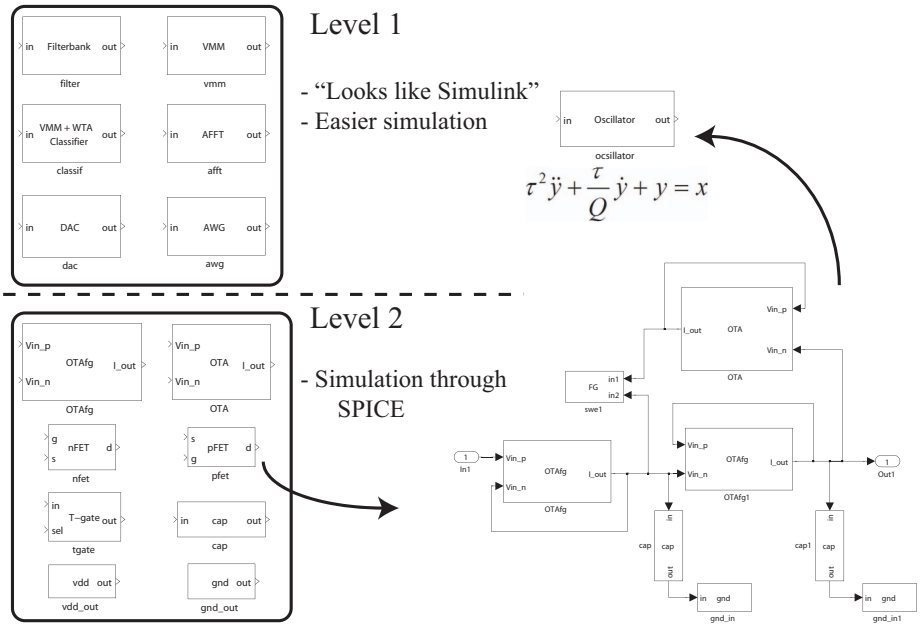

Fig. 4: To accommodate users with varying expertise, we provide multi-level libraries. The Level 1 library is meant for the highest level blocks. These blocks will include functions with which a typical DSP engineer will feel comfortable designing, such as filterbanks, vector-matrix multipliers, classifiers, analog FFT, DACs, and arbitrary waveform generators. The level 2 library contains the low-level blocks, such as the FPAA CAB elements, and is meant for experienced circuit designers. The right side shows the design cycle of a second-order section oscillator. The analog engineer designs with the Level 2 blocks, simulates with SPICE, tests on the FPAA, then creates a new Level 1 block.

common analog practice, Simulink operates in voltage mode and cannot have two outputs drive a line. Therefore, we have used a FG switch with two "inputs" to short the two nets. This results in a legitimate circuit that will simulate in SPICE and operate in silicon; it will not, however, simulate in Simulink.

To make this block useful to system designers, we abstract it to the high-level block shown in the top right of the figure. Here, we have expressed the eight-element circuit as a simple second-order differential equation [2]. This equation is very easy for Simulink to simulate. In this expression, the user would specify the time constant (τ) and the Q. These system parameters will be translated into physical parameters without the user's involvement. If we use equal capacitors and G_m for the forward FG-OTAs, we get $\tau = C/G_m$. With G_m set, we get $Q = 1/\left(2 - G_{mFB}/G_m\right)$, where G_{mFB} is the transconductance of the feedback OTA.

With the equation set, we just need to define the signal dimension. There is no need to add any conversion stages because this block is already voltage in and out. This block can be arrayed by allowing the user to input two n-element vectors, one for τ and Q. The block can automatically set its input and output ports based on the size of the parameter array. The resulting system will have n oscillator circuits in parallel, each programmed with the elements of the parameter array.

IV. Conclusion

In this paper, we have addressed the major challenges in developing an abstract analog system design framework. We discussed the trade-offs involved in fixing the interface mode between blocks, the conventions for vectorized nets, and methods for automated biasing. We also illustrated the process flow with a bottom-up example of an analog oscillator. With the development of large-scale FPAAs, the size and complexity of analog systems requires these high-level synthesis tools.

References

[1] P. Hasler and D. Anderson, "Cooperative analog-digital signal processing," in *IEEE ICASSP*, 2002, pp. 3972 – 3975.

[2] Carver Mead, *Analog VLSI and Neural Systems*, Addison Wesley, 1989.

[3] P. Hasler, "Low-power programmable signal processing," in *Int. Workshop on System-on-Chip for Real-Time Appl.*, 2005, pp. 413 – 418.

[4] G. Frantz, "Digital signal processor trends," *IEEE Micro*, vol. 20, no. 6, pp. 52 – 59, 2000.

[5] B. Murmann, C. Vogel, and H. Koeppl, "Digitally enhanced analog circuits: System aspects," in *IEEE ISCAS*, 2008, pp. 560 – 563.

[6] R. Robucci, K. Leung, J. Gray, J. Romberg, P. Hasler, and D. Anderson, "Compressive sensing on a cmos separable transform image sensor," in *IEEE ICASSP*, 2008, pp. 5125 – 5128.

[7] S. Peng et al., "A programmable analog radial-basis-function based classifier," in *IEEE ICASSP*, 2008, pp. 1425 – 1428.

[8] A. Basu et al., "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781 – 1794, 2010.

[9] C. Schlottmann and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation," *IEEE JETCAS*, vol. 1, no. 3, pp. 403 – 411, 2011.

[10] S. Suh, A. Basu, C. Schlottmann, P. Hasler, and J. Barry, "Low-power discrete fourier transform for ofdm: A programmable analog approach," *IEEE Trans. Circuits Syst. I*, vol. 58, pp. 290 – 298, 2011.

[11] C. Schlottmann et al., "A high-level simulink-based tool for fpaa configuration," *IEEE TVLSI*, vol. 20, no. 1, pp. 10 – 18, 2012.

[12] C. Schlottmann, C. Petre, and P. Hasler, "A high-level simulink-based tool for fpaa configuration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 10 – 18, 2012.

[13] P. Hasler, C. Schlottmann, and S. Koziol, "Fpaa chips and tools as the center of an design-based analog systems education," in *IEEE Int. Conf. Microelectronic Systems Education (MSE)*, 2011, pp. 47 – 51.