

# COOPERATIVE ANALOG-DIGITAL SIGNAL PROCESSING

Paul Hasler and David V. Anderson

Georgia Institute of Technology, Electrical and Computer Engineering, Atlanta, GA 30332  
phasler@ece.gatech.edu, dva@ece.gatech.edu

## ABSTRACT

We introduce the concept of cooperative analog-digital signal processing, and its implications on real-time signal processing functions. We discuss some of the trade-offs between performing operations in the analog computing circuit and digital computing circuits, and a general framework for "cooperative analog/digital signal processing systems" is presented. A key aspect of the cooperative analog/digital signal processing systems is the fact that new advances in analog VLSI circuits make it possible to develop advanced analog signal processing systems with programmable elements.

## 1. DEFINITION OF COOPERATIVE ANALOG-DIGITAL SIGNAL PROCESSING

New advances in analog VLSI circuits have made it possible to perform operations that more closely reflect those done in DSP applications, or that are desired in future DSP applications [1, 2, 3, 4, 5, 6, 7]. Further, analog circuits and systems can be *programmable*, reconfigurable, adaptive, and at a density comparable to digital memories (for example, 100,000+ multipliers on a single chip) [8, 9, 10, 11, 5]. Typically, one does not think of analog and programmability together—analog circuits are primarily for preamplifiers, and programmability has been exclusively in the domain of digital processing. Therefore, one might wonder if we have both digital and analog signal processing available, how does one choose a particular solution for a given application. The focus of this paper, and of this special session, is to address this question. This paper describes an effort to create cooperative analog/digital signal processing systems that benefit from the advantages of both types of systems to make something better than the sum of its parts.

We define cooperative analog-digital signal processing (CADSP) as looking at the issues of using combinations of programmable analog signal processing and digital signal processing techniques for real-world processing. Neither analog signal processing or digital signal processing can exist in current technologies without the other; that is, real-world signals are analog while much of the modern control and communication is digital. In the end, the question is where to partition the analog-digital boundary, as shown in Figure 1, to enhance the overall functionality of a system by utilizing analog/digital computations in mutually beneficial way. CADSP allows more freedom of movement for the partition between the analog and digital computation. CADSP is a superset of mixed-signal research in that it focuses heavily on algorithms as well as circuit implementation. By adding functionality to our analog systems, we enhance the capabilities of the controlling digital system, and therefore, the entire product under consideration.

A full discussion of this partition problem can and will encompass several research papers. The range of applications for these approaches reaches from auditory and speech processing,

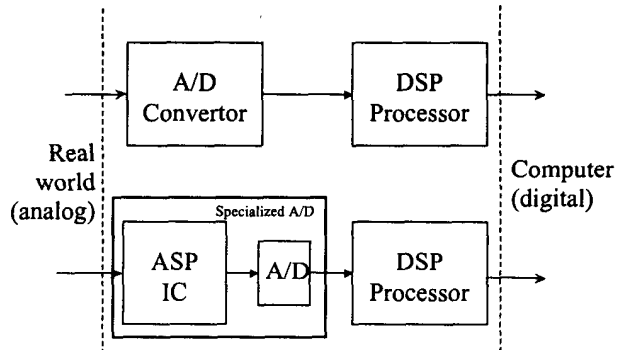
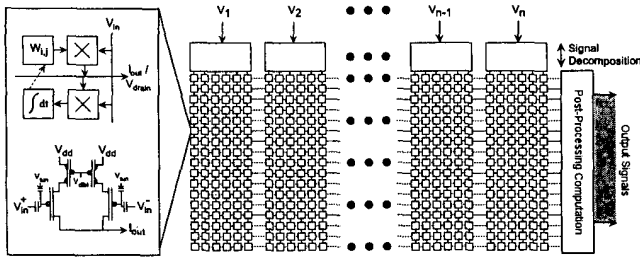


Fig. 1. Illustration of the tradeoffs in cooperative analog/digital signal processing. We assume the typical model of signals coming from real-world sensors, which are analog in nature, that need to be utilized by digital computers. The inverse problem, digital signals going to real-world actuators, is similar in nature. One approach is to put an analog-to-digital (A/D) converter as close to the sensor signals as possible, and allow the remainder of the computations to be performed digitally. An alternate approach is to perform some of the computations using analog signal processing, requiring simpler A/D converters, and reducing the computational load of resulting digital processors. One could group this analog computation and A/D converter as a specialized A/D converter that gives more refined information (Fourier coefficients, phonemes, etc.) than a literal map of the incoming signal. The question of where to put this boundary line strongly depends upon the particular requirements of an application.

to beam-forming, multidimensional signal processing, and radar computations, communications processing, and image processing and recognition. The following sections will begin to discuss the resulting capabilities, issues, and trade-offs as well as to discuss the implications on power dissipation, computational throughput, and engineering design time. Section II discusses the current technological environment and advances that make a combination of analog and digital signal processing feasible. Section III presents an overview of the capabilities of analog signal processing. Section IV presents an overview discussion of the comparative cost of resolution for a given application. The corresponding papers in this session will address one or more of these issues in more detail.

## 2. THE CASE FOR COMBINING ANALOG AND DIGITAL SIGNAL PROCESSING

One might wonder why introduce analog signal processing, since the current framework of immediately digitizing the incoming signal, illustrated as the top half in Fig. 1, seems to be working well in current practice, primarily being driven by transistor scaling



**Fig. 2.** Architectural layout of a simple analog computing array utilizing floating-gate memory elements—a typical system is an array of floating-gate computing elements, surrounded by input circuitry to precondition or decompose the incoming sensor signals, and surrounded by output circuitry to postprocess the array outputs. Additional circuitry is used to individually program each analog floating-gate element.

and flexibility of programmability. Current trends are, and have been, finding unique challenges that provide an opportunity for a new perspective. Additionally, analog VLSI advances have quietly made analog signal processing capable of similar programmable and adaptive operations to DSP.

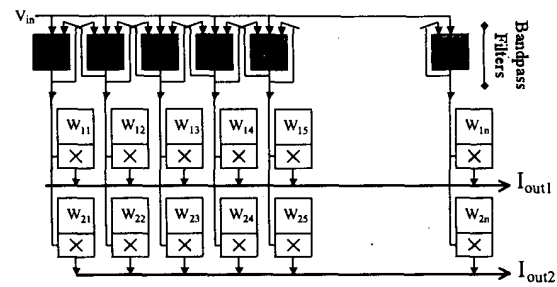
Several factors contribute to the case for using analog signal processing in conjunction with digital systems are

1. power consumption / efficiency,
2. A/D converter requirements,
3. size constraints, and
4. problem “fit” (this is discussed briefly in section 3).

Power consumption in DSP microprocessors, as measured in mW/MIPS, has been decreasing by half about every 18 months—a phenomena known as Gene’s law [12]. This has been keeping pace with Moore’s law and has helped make possible the increasing proliferation of portable electronics. Even so, device functionality, and the amount of signal processing, is often primarily constrained by a fixed power budget. A custom analog approach can often achieve an increased efficiency (= Bandwidth \* Power) of a factor of  $10^4$  over a custom digital approach. If Gene’s law continues to hold, digital systems will catch up to current analog efficiencies within about 20 years! Therefore, migrating some applications to analog processing could provide up to a 20 year jump in functionality relative to a purely digital roadmap.

Analog to digital (A/D) converters also can be a limiting factor in signal processing systems and A/D requirements are becoming an increasingly large part of the system design constraints. The system demands on many current systems require very high resolution / high performance A/D converters; the resulting A/D converter block is often consuming a large fraction of the power budget, as well as system design time. While digital processor efficiency may be increasing rapidly, partially as a result of transistor scaling; scaling is not helping as much for A/D converters. A/D converters have roughly been increasing resolution at 1.5 bits / 5 years at the same performance, and quickly running into additional physical limits which might further slow this progress.<sup>1</sup> In many problems the need for enormous speed and resolution originates in recovering “low-information” signals over a wide dynamic range or in a noisy background (e.g. CMOS imaging, software radio).

<sup>1</sup>The A/D converter limitations on a power constrained systems is somewhat similar to the problems generated as processor speeds progressed much faster than memory speeds in recent years.



**Fig. 3.** Top level picture of our Programmable Analog Fourier Processor. We separate the signal into frequency bands not by computing a DFT algorithm, but by a series of band-pass filters. We divide our frequency space exponentially, instead of linearly as in typical DFT algorithm. One current implementation for auditory and low MHz filtering. Here we choose a Fourier-transform like basis function; therefore the function is similar to DSP filtering using DFT and inverse-DFT stages.

By utilizing analog signal processing at the front-end, we can significantly reduce the A/D converter complexity, and the overall system complexity.

Circuit size constraints also favor analog VLSI circuits in many cases. As will be discussed more in the next section, it is often possible to take advantage of device physics to perform complex operations with only a very few transistors. For example, an analog multiplier can store the coefficient and perform the multiplication using only as many transistors as would be needed to store a four bit coefficient in digital memory.

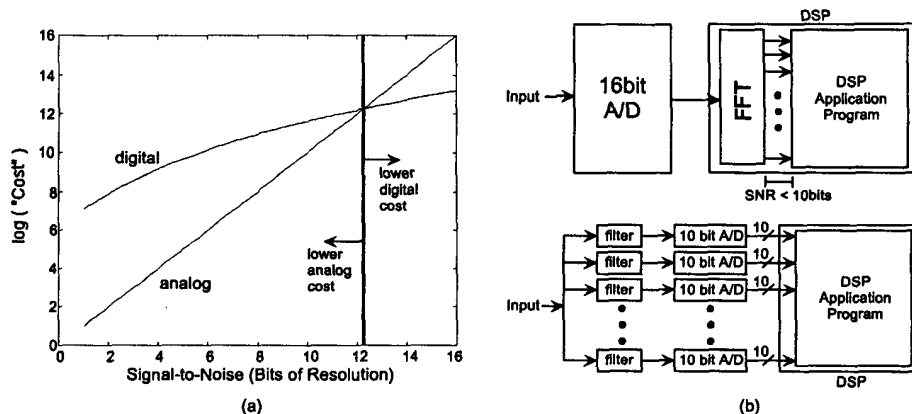
### 3. ANALOG SIGNAL PROCESSING CAPABILITIES

Before discussing trade-offs between modern analog and digital signal processing, we will summarize the capability of analog signal processing, particularly in an integrated circuit context. Many analog techniques are orders of magnitude more efficient, defined as the product of speed and power dissipation, than their digital counterparts [13]. The greatest efficiency occurs when matching physics of problem to physics of silicon medium. This increase in efficiency can be used for reducing the power efficiency of a given problem, or addressing computational problems that are considered *intractable* by the current digital roadmap.

Analog signal processing is not limited to single-input-single-output computations, but naturally conforms to many parallel matrix-vector computations [5, 14]. Silicon IC chips are two-dimensional, and therefore are subject to various constraints on this surface, similar to arrays of digital processors [15]. We can represent a matrix of stored values, input a vector of signals, and output a vector of signals. We find that vector-matrix multiplication is straight forward, where matrix-matrix multiplication is achievable if we can break up the computation into a sequence of matrix-vector multiplications (an  $O(n)$  problem). Further, we would find computing an eigenvector of a matrix very difficult, but computing an eigenvector of a covariance matrix straightforward by using algorithms that involve only matrix-vector multiplications

Because these processing elements are arrays of single memory elements, the resulting structure resembles a digital memory structure, except that in a single column memory access, we are computing at least a matrix-vector multiplication. Unlike digital

**Fig. 4.** Guidelines on using analog or digital signal processing depending upon required resolution (Signal-to-noise). (a) As discussed in several elsewhere [16], the computation cost of digital computation varies linearly with the required bits of resolution, while, the computation cost of digital computation varies exponentially with the required bits of resolution. This threshold is typically between 8bits to 14bits, depending upon the particular application. (b) An example comparison looking at the resulting SNR for two approaches for a particular applications: one case is a purely DSP solution, and the second case is a combined analog-digital solution. In the end, either approach would give similar amount of information at each output channel



memory, each cell acts as a multiplier that multiplies the analog input signal to that cell by an analog value stored in a floating gate. By performing the computation in the memory cells themselves we avoid the through-put bottlenecks found in most signal processing systems. Typically, these computing memory cells may be accessed individually (for readout or programming) or they may be used for full parallel computation within the array (as in matrix-vector multiplication or adaptation). Further, only one memory access per incoming sample is required, or in other words, the system only needs to operate at the incoming data speed (maximum input frequency); therefore, reducing requirements on overall system design.

Once the architecture is built, the size and power dissipation of these computing memory elements is critical, because we want as many elements as physically possible. The enabling technology of this approach is a floating-gate circuit technology that allows for simultaneous storage, computation, and programming in one or two single-transistor EEPROM cells [9, 17, 18]. Figure 2 shows a general block-diagram of a floating-gate computing array. This technology can be integrated in a standard digital CMOS process or in standard double-poly CMOS processes. We call these resulting computational memory arrays as high density analog computing arrays (referred to as computing arrays). The analog computing arrays are each capable of computing a product between a stored weight and an input. In addition, each of the cells can adapt due to input signal correlations and each cell allows for programming that does not affect the computation. Therefore, the array provides full parallel computation with the same circuit complexity and power dissipation as the digital memory needed to store this array of digital coefficients at 4-bit accuracy (two transistor cell). In addition, this approach offers promise to incorporate general analog signal processing blocks into FPGA and standard ASIC technologies.

Figure 3 shows the top-level description of the on-chip programmable analog filter concept [5, 19, 20]. Each tap consists of two pieces, one piece that projects the signal onto an analog basis function, and a second piece that multiplies the result by a stored weight. This computation is analogous to building a filter using an FFT at each incoming sample, weighting the resulting spectrum, and computing an IFFT. The output is a current that is placed onto

a common interconnection line. Voltage-mode band-pass filters are used to split the input signal into several time-dependent basis functions. By programming floating-gate elements on the filters, the corner frequencies can be arbitrarily spaced through audio and MHz frequency range (e.g. linear or exponential spacing). With one processor we can output multiple band weighted versions of the original signal.

#### 4. SIGNAL-TO-NOISE VERSUS COST

Even if analog signal processing is capable of several important functions, and is programmable, the primary question is the effective resolution of these computing systems. The related question is identifying the cost of computation at a particular resolution. Figure 4a shows a typical plot of signal-to-noise as bits of resolution versus the net cost [16]. One gets similar results when computing cost using a wide range of metrics involving area, power dissipation, computational delay, required tools, expenses associated with the design and manufacture, and design time. The computation cost of digital computation varies linearly with the required bits of resolution, while, the computation cost of digital computation varies exponentially with the required bits of resolution. As a result, computation requiring less resolution than a threshold is less expensive for analog computation, and computation requiring more resolution than a threshold is less expensive for digital computation. One careful study by Sarpeskar [16], showed that analog computation has significant advantages if the resolution of the incoming information is not sufficiently high, typically 10 bits or less. These concepts argue for analog implementations for many real-time sensor signal-processing/control problems. Of the two types of digital number formats, the analog computing circuits discussed behave more like floating point systems than fixed point systems in terms of the resolution.

The key in looking at the necessary resolution for either the analog or digital signal processing parts depends heavily on the amount of the incoming information and resolution needed to represent it. Figure 4b shows an example comparing how one might apply these results. One common signal processing step with incoming sensor data is taking an FFT, or equivalent Fourier based algorithm. For DSP computation, we would require a 16 bit A/D

converter to get some output channels at 10bit resolution. For ASP computation, we would require a bank of band-pass filters with 10bits of Signal-to-noise ratio coupled with a bank (or multiplexed) 10 bit A/D converter to get the output channels at 10bit resolution. Both analog systems have similar design complexity, because the design complexity of a 16 bit A/D converter is exponentially harder than the design complexity of a single or multiple 10bit A/D converters. These computations are transparent (in resolution) to the engineers developing the remainder of the algorithm, and therefore trade-offs could be made at these levels. Modeling analog signal processing resolution, typically measured in signal-to-noise ratio (SNR) must consider the particular circuit effects and continuous-time signal processing to get an accurate estimate. Simply treating analog components as fixed-point arithmetic with finite register effects will always underestimate the SNR of actual computation.

## 5. CONCLUSIONS: APPROACHING ANALOG-DIGITAL DESIGNS

With these options available, where in practice do we partition a system to use analog signal processing and a digital signal processing, and at what complexity at each stage? One should envision this process as either starting from a custom IC development, or from a board / single chip (parts already in hand) with analog FPGA (FPAAs), digital FPGA(s) + classical DSP processor, where we can turn power completely off to a section if desired. We start from typically a DSP algorithm, since digital approaches are often the starting technology and will be interfaced to digital approaches, or other sources of inspiration (e.g. neurobiology). Once we have chosen an algorithm, we typically proceed in the following framework:

- Is the analog signal processing available for the algorithms of interest?
- Most evaluate cost for the initial framework: Resolution, Power budget, Specifications, time to market
- Signal-Processing (e.g. MATLAB) level simulation of analog-digital signal processing system to optimize system performance.

Often several iterations through these steps are necessary.

The following papers in this session present several interesting particular studies looking at one or more aspects of this process. The papers will present opinions and guidelines for addressing this question based upon case studies and experimental IC characterization.

## 6. REFERENCES

- [1] Carver Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA, 1989.
- [2] K. Boahen and A. Andreou, "A contrast-sensitive retina with reciprocal synapses," in *Advances in Neural Information Processing Systems 4*, J.E. Moody, Ed. Morgan Kaufman Publishers, San Mateo, CA, 1991.
- [3] L. Watts, D.A. Kerns, and R.F. Lyon, "Improved implementation of the silicon cochlea," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 5, pp. 692–700, 1992.
- [4] J. Lazzaro and C. A. Mead, "A winner-take-all circuit in  $O(n)$  complexity," in *Advances in Neural Information Processing Systems 1*, Gerald Tesauro and David S. Touretzky, Eds., pp. 817–824. MIT Press, Cambridge, MA, 1989.
- [5] Matt Kucic, Paul Hasler, Jeff Dugger, and David V. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *2001 Conference on Advanced Research in VLSI*, Erik Brunvand and Chris Myers, Eds. IEEE Computer Society, March 2001, pp. 148–162.
- [6] Asad A. Abidi G. Tyson Tuttle, Siavask Fallahi, "An 8b cmos vector a/d converter," in *Proceedings of the IEEE International Solid State Circuits Conference*, Monterey, CA, 1993, pp. 257–259.
- [7] Jeremy Lubkin and Gert Cauwenberghs, "A micropower learning vector quantizer for parallel analog-to-digital data compression," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Monterey, CA, 1998, vol. III, pp. 58–61.
- [8] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network with 10240 'floating gate' synapses," in *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C., 1989, vol. II, pp. 191–196.
- [9] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7*, Gerald Tesauro, David S. Touretzky, and Todd K. Leen, Eds., pp. 817–824. MIT Press, Cambridge, MA, 1995.
- [10] Paul Hasler, Bradley A. Minch, and Chris Diorio, "Floating-gate devices: They are not just for digital memories anymore," in *IEEE International Symposium on Circuits and Systems*, Orlando, Florida, 1999, vol. II, pp. 399–391.
- [11] P. Hasler and T.S. Lande, "Special issue on floating-gate devices, circuits, and systems," *IEEE Journal of Circuits and Systems*, vol. 48, no. 1, Jan. 2001.
- [12] Gene Franz, "Digital signal processor trends," *IEEE Micro*, vol. 20, no. 6, pp. 52–59, Nov–Dec 2000.
- [13] Carver A. Mead, "Neuromorphic electronic systems," *IEEE Proceedings*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
- [14] Gert Cauwenberghs, *Learning in Silicon*, Kluwer Academic, 1999.
- [15] S.Y. Kung, *VLSI array processors*, Prentice Hall, Englewood Cliffs, N.J., 1988.
- [16] Rahul Sarpeshkar, *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*, PhD thesis, California Institute of Technology, Pasadena, CA, 1997.
- [17] Paul Hasler, Bradley A. Minch, Jeff Dugger, and Chris Diorio, "Adaptive circuits and synapses using pfet floating-gate devices," in *Learning in Silicon*, Gert Cauwenberghs, Ed., pp. 33–65. Kluwer Academic, 1999.
- [18] Paul Hasler and Bradley A. Minch, *Floating-Gate Devices, Circuits, and Systems*, IEEE Press, 2002.
- [19] P. Hasler, M. Kucic, and B. A. Minch, "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *Midwest Conference on Circuits and Systems*, Las Cruces, NM, 1999.
- [20] Matt Kucic, AiChen Low, Paul Hasler, and Joe Neff, "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 90–99, Jan. 2001.