Reconfigurable Analog Signal Processing for Wireless Sensor Networks

Brandon M. Kelly, Brandon Rumberg, David W. Graham, and Vinod Kulathumani
Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506
{bkelly6, brumberg}@mix.wvu.edu and {david.graham, vinod.kulathumani}@mail.wvu.edu

***Abstract*—The limited power budgets of sensor networks necessitate in-network pre-processing to reduce communication overhead. The low power consumption of analog signal processing (ASP) is well-suited for this task. However, adoption of ASP is restrained by the longer design time relative to reconfigurable/reprogrammable digital processing. Our solution is to enable ASP reconfiguration through the use of a field-programmable analog array (FPAA), which allows wireless sensor network developers to rapidly prototype and test ASP designs. In this paper, we present an FPAA designed for use in wireless sensor networks, and we describe its incorporation and use within a sensor node.**

I. Introduction

Wireless sensor networks (WSNs) are capable of a myriad of tasks, from monitoring critical infrastructure such as bridges to monitoring a person's vital signs in biomedical applications. However, their deployment is impractical for many applications due to their limited power budget, which is mostly spent on communication [1], [2]. In-network pre-processing can help reduce this communication overhead.

As we demonstrated in [3], analog signal processing (ASP) is a form of in-network pre-processing that can be used to process a signal locally, provide event detection for wake-up scenarios, and more. It is a compelling choice to use analog processing in sensor networks due to the analog nature of real-world signals and due to the 20 year leap that analog electronics have been shown to have over digital systems in terms of performance-per-power consumed [4].

Analog sensor interfaces in WSNs tend to be application-specific and are not necessarily capable of operating in a range of other applications. Additionally, ASP design is lengthy, compared to digital systems which can make use of programmable and reconfigurable systems. Therefore, incorporation of an ASP front end is typically more costly in terms of time to market and also development costs.

To overcome these challenges, we recommend improving an ASP's implementation by making it reconfigurable. A reconfigurable architecture would allow a single ASP IC to be used for a variety of applications and to be updated in the field as its application is redefined. To that end, in this paper we have expanded upon our previous work [3] by including a field-programmable analog array (FPAA) in a manner similar to [5], [6]. Additionally, we have improved upon our previous architecture to allow for more complex processing, and thus more discriminating detection of sensed phenomena.

This material is based upon work supported by the National Science Foundation under Award No. 1148815.

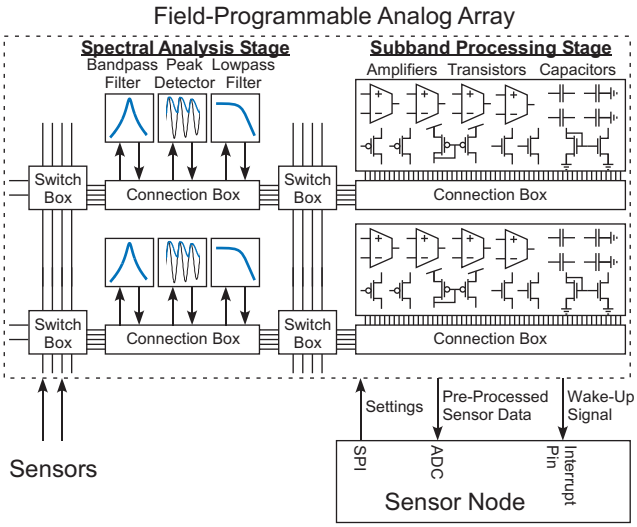

Fig. 1. A sensor node equipped with a field-programmable analog array (FPAA). The FPAA can be reconfigured in run-time to perform event detection and pre-processing at a power consumption that is significantly lower than the power that would be consumed by the mote's built-in digital systems.

II. FPAA Architecture

Previous FPAA designs have already demonstrated the ability to synthesize complex analog circuitry [7]; what we would like to establish is their viability in embedded systems such as WSNs. More specifically, we would like to establish an FPAA's use within energy-constrained systems that monitor phenomena such as audio, vibration, and motion, which have sufficiently high bandwidths (>100Hz) to challenge the throughput of typical WSNs.

The FPAA we present here for use in WSNs consists of four computational analog blocks (CABs), including two for spectral analysis and two for subband processing (Figs. 1 and 2). The spectral-analysis stage consists of two bandpass filters in the form of capacitively-coupled current conveyors (C^4 filters), two envelope detectors, two adaptive-time-constant filters for suppressing envelope detector ripple, and two buffers. These subblocks all have tunable biases, allowing the user to perform several common types of analysis on a

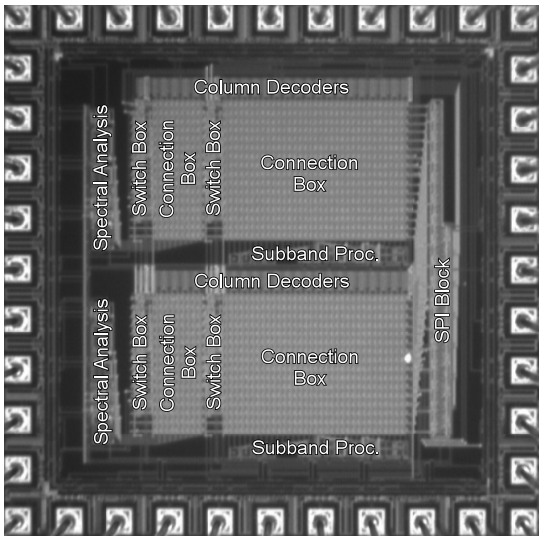

Fig. 2. Die photograph of the FPAA fabricated in a standard $0.5\mu m$ CMOS process available through MOSIS. The FPAA is 2.25mm^2.

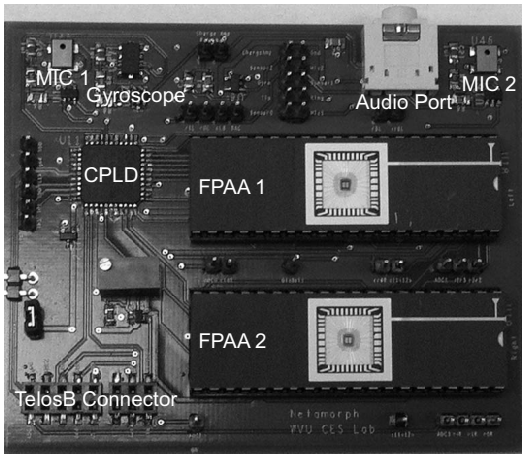

Fig. 3. PCB for interfacing the FPAA to a WSN sensor node. This board incorporates a variety of sensors, a CPLD, a DAC (on the underside of the PCB), and a socket for connecting a TelosB mote.

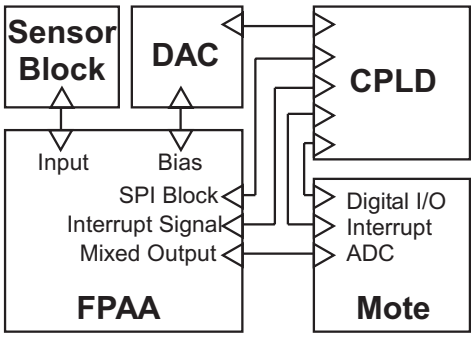

Fig. 4. A high-level schematic of the PCB for interfacing sensors, the FPAA, and a WSN mote.

range of signals before further processing. Further processing takes place in the subband processing CABs, which provide the user access to elements of a smaller granularity, including operational transconductance amplifiers (OTAs), current mirrors, individual transistors, and capacitors.

A notable feature of this FPAA architecture is that, because it concentrates on relatively high-frequency phenomena such as certain types of simple harmonic motion, it provides both general analog computing tasks, and it is also suitable for parallelized processing. The latter feature stems from the fact that the subbands of the architecture can be configured to perform identically, and the architecture itself could easily be scaled up to further this task.

In our FPAA, reconfiguration is achieved via programmable switches in the connection box (which is used for intra-CAB routing) and the switch box (which is used for inter-CAB routing). The connection box consists of a full-crossbar configuration for flexible local routing. In the switch box, we implemented a variety of connection types (such as crossbar, crossover, and four-way switch points) to evaluate their value within sensor networks. To facilitate ease of integration with a sensor node, we implemented these switches using SRAM-controlled transmission gates. Each switch had an SRAM memory cell that set it to "on" or "off." To load values, we used a row-by-row method that would load the state of all 16 switches in a given row. The configuration was written into the SRAM array using an on-chip serial peripheral interface (SPI). In total, the FPAA had 1436 switches, with the potential to route 40 unique nets.

III. Mote Interfacing

Our goal with this design was to integrate the FPAA into a WSN sensor node in a way that would enable us to easily monitor a range of phenomena. To that end, we created a printed circuit board (PCB), as shown in Figs. 3 and 4, that includes a variety of sensors, two FPAAs to enable scalability, a TelosB mote connector, a digital-to-analog converter (DAC) for providing bias voltages, as well as a complex programmable logic device (CPLD). For the sensors, we chose to focus on the relatively high-frequency phenomena that are traditionally very taxing on a WSNs power budget, including motion, audio signals, and various forms of simple harmonic motion. To monitor these phenomena, we equipped the board with a gyroscope, two microphones placed at opposite ends of the board to enable directional sensing, and a mini-stereo port to enable future expansion.

By including two FPAAs on the same board, we were able to effectively scale up our architecture, overcoming our IC space limitation, and build more sophisticated ASP designs. The FPAAs include SPI blocks that can be programmed directly through the attached TelosB mote's general purpose input/output (GPIO) pins. The on-board CPLD was used to minimize the number of TelosB pins that were used for digital I/O, thus freeing up more pins to be configured as ADCs. Additionally, the CPLD was used to define even more complex wake-up events. We also simplified setting individual bias points for all of the ASP blocks by including DACs which can be set and adjusted directly by the mote.

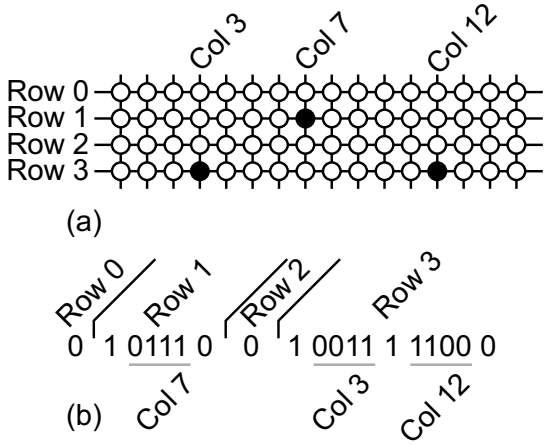

Fig. 5. (a) Example of a configuration of switches. (b) Implementation of how this configuration would be transmitted using our compression scheme.

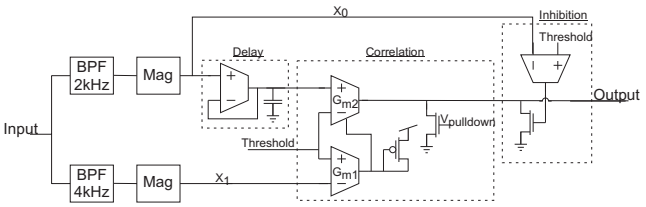

Fig. 6. Top-level schematic of the circuit synthesized in the FPAA to demonstrate its spectral analysis capabilities. The circuit detects portions of the signal where the frequency content rises in the 2-4kHz range. The "Correlation" stage detects the simultaneous presence of content in the high-frequency band and the delayed low-frequency band. The "Inhibition" stage nulls the output when content is present in the low-frequency band to avoid triggering on wideband signals. G_{m2} is biased by the gate of the attached pFET while the output current of G_{m1} is mirrored through the diode connected FET. Also note that $V_{pulldown}$ is a constant bias used to weakly pull down the output when G_{m2} is shutoff by G_{m1}.

A. User Interface

We developed a software user interface to aid in the reconfiguration and tuning of the FPAA. This user interface simplifies the process of synthesizing the analog circuits on the FPAA to aid WSN designers who may not have the circuit-level expertise usually required to construct an ASP. The current implementation of the user interface allows users to construct individual ASP blocks and see the routing that the device will use to connect them. Once the user is satisfied with the design, the configuration, which consists of the switch settings and the DAC bias values, is converted to a header file by a Matlab script. The updated header file is then uploaded to the base-station mote running TinyOS, and then wirelessly transmitted to the remote nodes. The remote node then applies the new configuration to its FPAA.

B. Compression

When designing an FPAA for use in a WSN, the size of the FPAA is a critical design choice. While it is desirable to have an FPAA that is large enough to create sophisticated ASP designs, care must be taken to minimize the overhead of delivering and storing large configuration files within the network. The naïve approach for handling configuration files is to simply transmit the raw bits that will eventually be shifted into the FPAA. For example, the configuration file for the switch configuration in Fig. 5(a) will consist of 64 bits, only three of which are "on," which implies that the configuration is redundant. If this method were scaled to large FPAAs that have 74,000 switches [7], for example, then significant energy would be wasted transmitting and receiving redundant bits.

To address this problem, we have developed a compression method that is inspired by entropy coding, but that is informed by our observations about typical FPAA configurations. Configuration files tend to be small. Therefore, traditional methods which utilize a codebook would have too much overhead (e.g. Huffman coding). Even when considering FPAAs of larger scale, ASP algorithms tend to have a parallel nature and are still amenable to compression. An example of this would be a large filter bank which utilizes the same operation in each subband. This identical operation means that the switch settings would be the same in all channels; therefore, it would only be necessary to transmit the settings for one channel and then apply those settings to the remaining channels.

Due to redundancies in the switching matrix, most rows tend to have no switch set. Therefore, we begin our row-by-row configuration scheme by delineating whether or not any switches are set within a given row. Only if a switch is set in the row do we specify the location of the switch within the row using a four bit identification number. An example compression is shown in Fig. 5(b), where the 64-bit configuration of Fig. 5(a) is successfully reduced to 19-bits. The size of the compressed configuration depends upon the number of "on" switches N_{on}, and is equal to $5N_{on}$+N_{rows}, where N_{rows} is the number of rows in the FPAA. We have determined experimentally that the energy for the mote to decode the configuration is 34.1μJ, while the reduction in transmitted data saves 3.5mJ.

IV. System Examples

To illustrate the functionality of FPAA-based ASP designs in WSNs, we connected our FPAA interface board to a TelosB mote and synthesized several signal-processing circuits on the FPAA. Each of these circuits could be used to generate wake-up signals to turn on the TelosB mote. In each scenario, all reconfiguration commands were sent over the radio through another TelosB mote, which acted as the base station.

The first system that we demonstrate is the ability of the FPAA to perform basic spectral analysis (Figs. 6 and 7). Here, the FPAA has been configured to analyze a signal's frequency content and detect a rising frequency in the 2-4kHz range. The signal is first filtered through parallel bandpass filters set to center frequencies of 2kHz and 4kHz. The lower-frequency signal (x_0) is then delayed. A cascade of OTAs computes the product of the delayed low-frequency signal with the instantaneous high-frequency signal (x_1), thus providing a measure of simultaneity, reminiscent of the motion-analysis system in [8]. To ensure that static wideband signals do not trigger the detector, the final portion of the circuit pulls the

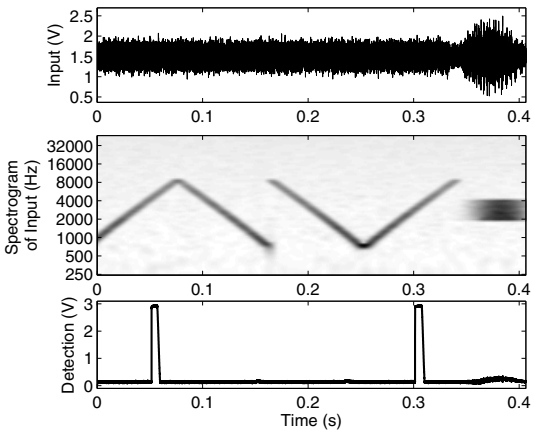

Fig. 7. Spectral analysis performed by the analog IC. (Top, Middle) The transient plot and the spectrogram plot of the input signal, respectively. The sinusoidal signal, which includes Gaussian noise, varies from 1kHz to 8kHz and concludes with a steady input of ten sine waves ranging from 2kHz to 4kHz. (Bottom) The output stage successfully detects portions of the signal where the frequency content rises in the 2-4kHz range.

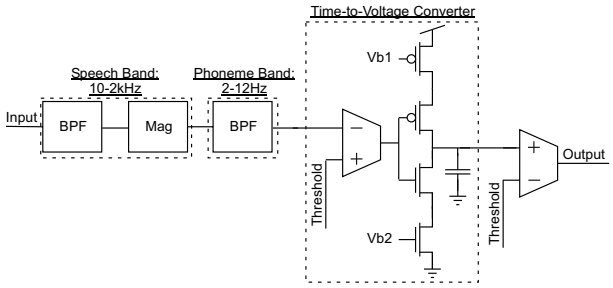

Fig. 8. Schematic of the implemented voice-activity detection algorithm. The device triggers an event when the amplitude modulation in the speech band occurs at a rate that is typical of speech.

output low when x_0 is high. The resulting output is high only when the x_1 and the delayed version of x_0 are high and the x_0 is not high. As a result, a pulse is generated when the signal is rising in the correct frequency range.

The next system demonstrates the ability of the FPAA to implement a voice-activity detector, based upon the scheme presented in [9] (Figs. 8 and 9). Audio signals were first passed through the spectral analysis CAB where they were filtered from 10 Hz to 2kHz using a bandpass filter. The envelope of this speech band was then found and passed through another bandpass filter, with corner frequencies at 2Hz and 12Hz corresponding to the phoneme band. The magnitude of the phoneme band was then used to trigger a time-to-voltage converter that would create a ramping voltage when the phoneme band exceeded a specified threshold. The time-to-voltage converter then triggers an event when this ramped voltage exceeded a threshold. While the inclusion of the time-to-voltage converter and the subsequent comparator stage may seem redundant at first, it allowed the device to operate in noisy, non-idealized conditions. The signal at each stage is shown in Fig. 8, and it is shown that the speech portion of the input signal was correctly identified in the presence of noise.

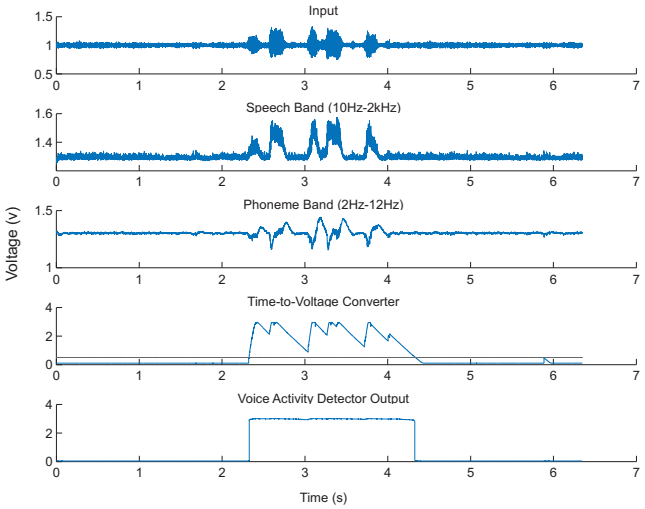

Fig. 9. Demonstration of the analysis of a signal throughout the voice-activity detection algorithm. For this test, the input was a male voice corrupted by noise from an airport environment, at a signal-to-noise ratio of 10dB.

The overall output can be used to identify to the rest of the sensor node that a signal of interest has been found.

V. Conclusions

In this work, we have presented our FPAA architecture which was designed for use in WSN systems. Also, we have shown how an FPAA can be easily integrated into a WSN system, including those which monitor phenomena of relatively high-frequency content. Finally, we demonstrated the system's ability to recreate complex ASP designs.

References

[1] S. N. Pakzad, G. L. Fenves, S. Kim, and D. E. Culler, "Design and implementation of scalable wireless sensor network for structural monitoring," *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 89–101, 2008.

[2] V. Raghunathan, C. Schurgers, and M. B. Srivatsava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, Mar. 2002.

[3] B. Rumberg, D. W. Graham, V. Kulathumani, and R. Fernandez, "Hibernets: Energy-efficient sensor networks using analog signal processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 321–334, 2011.

[4] S. Ravindran, P. Smith, D. W. Graham, V. Duangudom, D. Anderson, and P. Hasler, "Towards low-power on-chip auditory processing," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 3, pp. 1082–1092, Jan. 2005.

[5] Y. Chen, C. M. Twigg, O. A. Sadik, and S. Tong, "A self-powered adaptive wireless sensor network for wastewater treatment plants," in *IEEE International Conference on Pervasive Computing and Communications Workshops*, March 2011, pp. 356–359.

[6] E. Mackensen and C. Muller, "Implementation of reconfigurable microsensor interfaces utilizing FPAAs," in *IEEE Sensors*, Nov. 2005, pp. 1064–1067.

[7] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. M. Twigg, and P. Hasler, "A floating-gate-based field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sept. 2010.

[8] R. R. Harrison and C. Koch, "A robust analog VLSI Reichardt motion sensor," *Analog Integrated Circuits and Signal Processing*, vol. 24, pp. 213–229, 2000.

[9] T. Delbrück, T. Koch, R. Berner, and H. Hermansky, "Fully integrated 500μw speech detection wake-up circuit," in *Proceedings of ISCAS*, 2010 2010, pp. 2015–2018.