A Remote FPAA System for Research and Education

Sahil Shah, Jennifer Hasler, Sihwan Kim, Ishan Lal, Matt Kagle, and Michelle Collins
Georgia Institute of Technology, Atlanta, GA 30332–250 USA E-mail:jennifer.hasler@ece.gatech.edu

***Abstract*—We present a novel remote test system, enabled by configurable analog–digital ICs to create a simple interface for a wide range of experiments, whether in research or educational directions. Our remote test system utilizes a nearly identical setup to the existing large-scale Field Programmable Analog Array (FPAA) toolset; a mixed-mode configurable system with a common digital interface (e.g. USB) enables a nearly seamless transition. The system overhead requirements are straightforward, requiring simple email handling, available over almost all network systems with no additional requirements. We present using the FPAA devices and baseline tool framework, present overview examples for the remote system.**

We present and demonstrate a novel remote test system, enabled by configurable analog–digital ICs to create a simple interface for a wide range of experiments. Our remote test system opens opportunities for a simple remote test infrastructure by utilizing a highly configurable large-scale Field Programmable Analog Array (FPAA) device [7] and high-level tool framework [8], as well as a straight-forward digital interfaces for the resulting experimental system. The system overhead requirements are straightforward, requiring simple email handling, available over almost all network systems with no additional requirements.

This paper discusses a novel remote test system, enabled by configurable analog–digital ICs to create a simple interface for a wide range of experiments. We have seen a wide range of previous remote test systems that have to spend considerable time developing their hand-tailored configurable system [1], [2], [3], [4], [5], [6]. Figure 1 remote test system using simple email handling, available over almost all network systems. Independent of the distance, the system enables users from anywhere we have an internet connection sufficient to send and receive email. This approach minimizes computer support setup and maintenance, relieving the pressure overworked computer support staff, particularly in cost-conscious academic environments, trying to keep pace to maintain a larger number of computing systems. With a single button click in the graphical tool, the system will email the resulting targeting code for the FPAA device to a server location, to be picked up by the remote system, that compiles, runs, and then emails back the target results.

This approach gives a simple digital peripheral using a standard interface (i.e. USB), enabling a small Internet of Things (IoT) block interfaced through an email system to an open-source design / control tool. The resulting controlling device, whether it be directly connected through this digital port (i.e. phone or tablet) or through a network, allowing minimal (linux) code for programming and operation for continuous data processing, enabling all features on the resulting system, including sensory / actuation devices connected to this remote platform. Whether remote or not, we see these concepts as an easy approach for networks of remote sensor nodes, potentially enabling real-time sensor processing that can send context aware results while requiring minimal support overhead for the approach. We see the opportunities both in academic, as well as research and industrial applications. Our novel open-source tool platform empowers the user to do seamless low-power analog-digital CoDesign in a single environment.

This technical platform enables collaborators in different areas to investigate items on a single platform; setting up a system is straight-forward and capable for multiple systems. Our novel open-source tool platform empowers the user to do seamless low-power analog-digital CoDesign in a single environment [8]. Our FPAA SoCs consist of an integrated processor, I/O peripherals, and a FPAA comprised of analog and digital components, although the approaches can be extended to other platforms. Our approach integrates multiple open-source tools, using Scilab and VPR [10], to develop a coherent user-friendly design flow, with a custom software toolkit that generates and implements high-level simulation and experimental measurement of the resulting hardware system. The ability to seamlessly move between tools designed for a board in hand as well as a remote test system has greatly improved the student's interest in using either hardware platform, as well as kept development complexity for such a complex laboratory course under control.

This approach gives a simple digital peripheral using a standard interface (i.e. USB), enabling a small Internet of Things (IoT) block interfaced through an email system to an open-source design / control tool. The resulting controlling device, whether it be directly connected through this digital port (i.e. phone or tablet) or through a network, can be a potentially simple OS enabling the resulting system, including sensory / actuation devices connected to this remote platform.

I. Remote Tool Framework and FPAA Overview

Figure 1 shows the framework for the remote system approach. We want the resulting structure to be as easy to use on both the user and remote server side, requiring minimal user maintenance, using a integrated user tool platform, and having as few location constrains as possible. Our modifications to the high level Xcos tools on the user side required extending the GUI interface, as well as updating the (python) code to enable

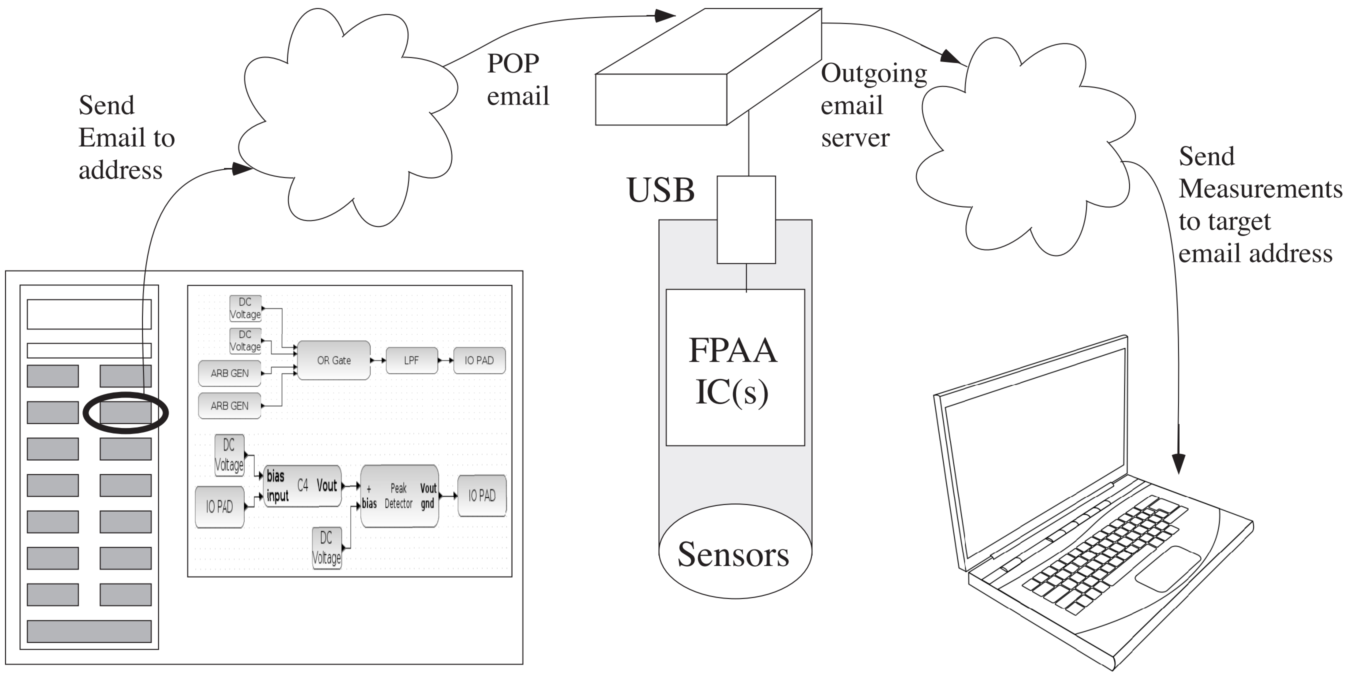

Fig. 1. Detailed flow for the remote test system implementation. The design toolset in scilab / Xcos allows the user to "send email" in addition to "program FPAA". When that option is chosen, the resulting file is sent, by email into the cloud. The resulting email is POP-ed off the server, the resulting programming files are extracted and executed, the resulting data measurement is performed on the device, and the results are sent back by email to the original sender. The user can directly use the results in Scilab or any other data analysis program to observe their data as well as complete their analysis. The resulting flow is enabled by having a highly configurable analog / mixed-mode system with a simple digital interface through USB, really enabling the connection as a typical digital peripheral.

emailing the resulting compiled FPAA targeting file out from the Virtual Machine (VM). The user will receive their resulting data to their email location as an attached representation of the measured results; the user can move this data into Scilab or another analysis / plotting tool of their choice. By using an email-based system, one would not expect real-time control occurring through the resulting email network.

We package the entire tool flow, illustrated in Fig. 1, from Scilab/Xcos, device library files, and compilation tools into a single Ubuntu 12.04 VM[1] that encapsulates the entire toolset that simply requires pressing one button to bring up the entire graphical working toolset. The approach allows us a given user to access and use this toolset on their remote computing devices with minimal effort, and we will assume this perspective as we dig deeper into the remote system approach. We could be designing with a single FPAA IC, multiple ICs, rack of boards, etc. as well as a set of other components that have some programmability. The high-level graphical tools enable a user to be able to try different approaches to optimize the system performance, allowing consideration of tradeoffs of power, system utilization, time to market, etc.

The tools output a single programming file, that is a combination of multiple smaller files compressed into a single structure, that is used for FG programming and SRAM memory setup for the SoC FPAA. The design tools can process the downloading of this file, as well as other devices (e.g. remote computer, tablet). The SoC FPAA devices enable Floating-Gate (FG) device programming entirely on the device as an input data stream, therefore the entire data stream, including μP code to execute programming, simply looks like a single stream of data to the system. The FPAA utilizes an open-source μP, embedded 16k $\times$ 16 SRAM for program and data memory, as well as the memory mapped registers for FG programming. We give the file definition in Table 1. We expect this structure will remain relatively stable across future generations; fortunately, each programming file is self contained for programming since it includes its own code and parameters for programming.

TABLE I
FORMAT AND SUBFILES FOR THE FPAA PROGRAMMING FILE

Function	Data Type	Core Files
erasing and initialization	Compiled (assembly)	tunnel_revtun_swc_CAB.elf switch_program.elf
measuring outputs	Compiled	voltage_meas.elf
input (e.g. DAC) data	data	input_vector
FG block	data	output_info
info (num, address)		switch_info target_info
switch list	data	
Course Prog T_{inj}	data	pulse_width_table
Fine prog V_d table	data	Vd_table_30mV

The remote server platform is also kept to require minimal resulting overhead. We utilize standard email servers to enable a relatively stable remote platform capable with nearly zero administrative overhead. The remote server will periodically check for email on the server, POP the resulting message from the server, check its control syntax, and have the object code ready for programming. Recent FPAA devices now enable

[1]available at http://users.ece.gatech.edu/phasler/FPAAtool/index.html

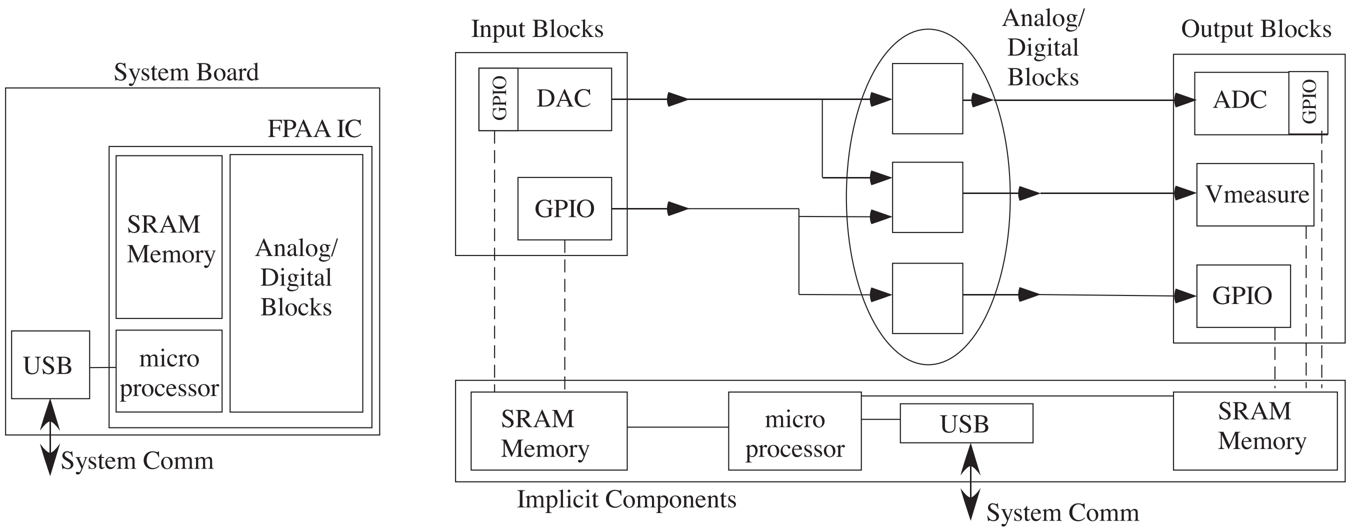

Fig. 2. Our approach is enabled through a single port of communication, through a USB port to a self contained programmable and configurable mixed-signal IC through its programmable interface. This simplicity results in understanding both the analog and digital capability of the FPAA IC, as well as the resulting input and output blocks for the system as well as the resulting system control, at the level allowed for the high-level tool framework (Xcos / Scilab). The entire IC is the computing system, and all components are part of the computation. A typical user will often use digital (GPIO) or analog (DAC through GPIO) input blocks, interfaced through the SRAM memory and μP control, and will often use digital as well as analog compiled ADC through GPIO or voltage measurement through slower and more accurate 14bit ADC. The tool framework compiles down the resulting analog, digital, and μP components, as well as the vector input into the system.

Floating-Gate (FG) device programming entirely on the device as an input data stream; therefore the entire data stream, including μP code to execute programming, simply looks like a single stream of data to the system. Encapsulating this entire structure in a single file requires small, unix-based code to communicate the file to be programmed. After the device is programmed and the input data is loaded into the processor, the IC proceeds to compute the resulting function, also storing data in local or in the remote server memory. The resulting output data results is pulled together and sent out by email to the host's chosen email address.

When one thinks of testing a configurable analog platform, one thinks of systems mostly having a single or a group of analog ICs, some controlled switches, and an infrastructure to connect to a computer to control the entire setup, as well as run the testing interface. When looking at a remote testing system, the testing interface layer requires further complication, depending on the particular system. In the case for a SoC FPAA devices like we utilize in this paper, interfacing requires communication through USB or potentially SPI ports, appearing to be a typical standard peripheralx. Such an approach enables a family of configurable hardware, utilizing a single configurable framework and tool infrastructure to control the resulting device. This difference in configuration provides the opportunity for empowering our remote test system, we have these examples for classroom use, research groups, as well as interested users. Integrating this approach into an existing tool framework keeps compatibility for a range of applications.

II. Remote User Interfacing Capability

Figure 2 shows the remote server was partially enabled by a simple digital (USB) interface, with simple interfacing between the FPAA IC (or multiple ICs) to the resulting USB infrastructure to the host device. The input data, output data, FG programming, and other control functions all move through a single standard digital USB interface; the device to the remote system looks like any other embedded, USB peripheral, where the tools handle a similar case whether the board is local to the tools or emailed to the remote server. We currently use a setup with no external pins; one could connect multiple devices, heterogeneous devices, and additional sensors, but to the external world, the device is still a simple USB connected device. The practical issue is understanding the particular interfacing options available on the FPAA, as well as the computation possible on the FPAA device. The USB interface is connected through serial interfaces on the device; in our case, we have the potential of a simple serial (8n1) debug interface.

Figure 2 shows the high-level blocks representation, similar to the corresponding Xcos diagram, both including computational blocks, as well as input and output blocks available and their interfacing into the μP / SRAM memory block. The arbitrary waveform generator uses data as a vector input representation versus time in the Scilab workspace (i.e. DAC devices) and packed with the programming file; the interface for the data is directly set by the high-level tools. A short list of potential on-chip interfacing options for moving data in and out of the μP / SRAM memory are:

- General Purpose Input / Output (GPIO) memory mapped digital registers
- hardware interrupts based on routed fabric signals
- signal DAC through memory mapped registers
- Compiled ADC coupled with FG elements in the fabric through GPIO registers

These approaches are all voltage-mode inputs and outputs, consistent with level=1 block definition [11] for system

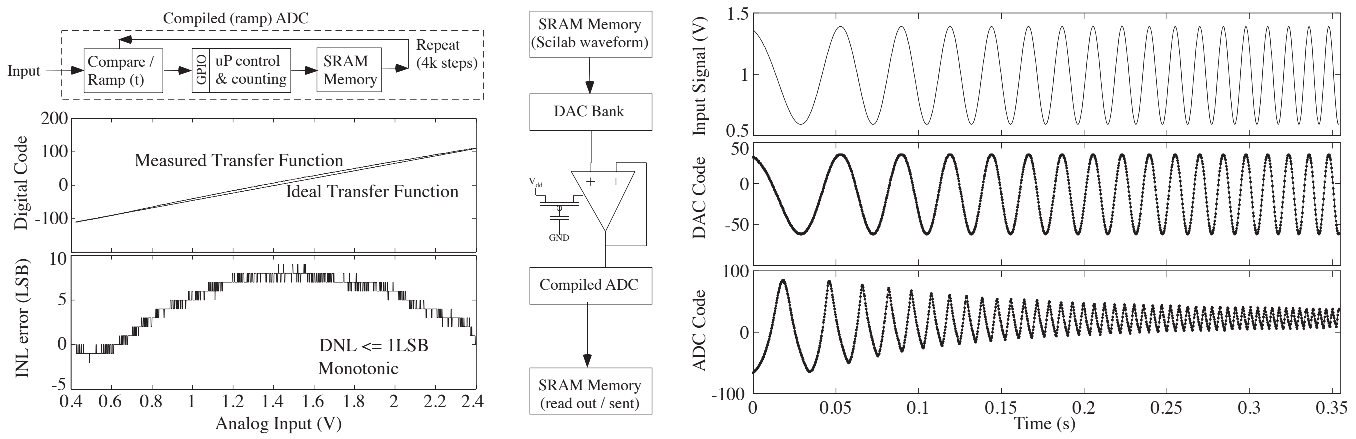

Fig. 3. Measurement of the compiled 8-bit ramp ADC used in the FPAA device. 4kSPS ADC for this 8-bit ramp conversion integrates the compare + ramp functions compile into a single CAB, routing the digital output bit to GPIO. This simple ADC polls for the digital bit to change while counting in the processor; more advanced blocks utilize interrupts and compiled counter blocks in the CLBs for the conversion. The ADC is not meant to be the highest precision component, but we have a very small, monotonic ADC that we can calibrate the resulting response as necessary. Further, we show a simple example of a simple compiled system for a first-order low-pass filter; we don't explicitly draw the load capacitance of the circuit (output going into the ADC), because we always have capacitance at a node, often from the routing infrastructure. We programmed the corner frequency of the LPF at 150Hz, and measured the resulting device by applying a linear chirp signal going from 25Hz to 250Hz, showing the resulting signal attenuation, as expected, as well as the resulting signals from the input 7-bit DAC as well as the 8-bit ramp-ADC (4kSPS). The output of the ADC inverts the resulting response from the original signal.

building. Further, blocks can have some assembly code (or completely assembly code) as part of the functionality, therefore would interact with memory, potentially, more directly. Where necessary, one can measure currents through compiled transimpedance amplifiers, switched capacitor network, and related circuit techniques. One key constraint on any algorithm is efficiently using the 16k x 16 SRAM space either to hold data, or buffer data coming through the serial communication from the host system running the remote interface.

One representative question would be the types of Analog-to-Digital Converter (ADC) blocks, and their applicability for a configurable architecture. An important question is what type of ADCs to compile in such a configurable system, particularly in a compiled system where we have a high density of FG devices available. One would not expect an architecture with lower latency than a pipelined 1-bit ADC architecture, primarily because if we had a smaller control loop, we likely would directly compile the computation in analog circuits. We expect these pipelined devices mostly for acquisition of data, often for circuit debugging opportunities. We expect that algorithmic converters, being related to the pipelined ADCs, would most likely win over successive-approximation ADCs because of not requiring the design of a separate DAC for the system. The approaches get closure on the most promising ADC IP blocks to compile down for these architectures.

Figure 3 shows a simple compiled ramp ADC converter to illustrate the flow from input SRAM memory data to the computed output SRAM memory data. The resulting simple 8bit ramp ADC illustrates using the digital infrastructure and μP to move analog signals into stored SRAM memory; the simplicity only requires part of a single CAB element while still giving reasonable monotonic performance with some curvature due to the nonideality of the current source element creating the ramp; we will show measurement examples at 4kSPS. Figure 3 also shows a complete computing loop for data through a board, as we see for a remote test, where we start with data loaded into SRAM as part of the programming structure, processes through one of multiple memory mapped 7bit DACs, through a first-order LPF block programmed with a corner frequency of 150Hz, through the above ADC block, to arrive back as stored solution vector in SRAM that is transmitted back to the user.

References

[1] Ananda Maiti et.al, "Merging Remote Laboratories and Enquiry-based Learning for STEM Education," *IJOE*, 2014.

[2] V.J. Harward et. al, "The iLab Shared Architecture: AWeb Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proceedings of the IEEE*, 2008.

[3] D. Lowe et. al, "Evolving Remote Laboratory Architectures to Leverage Emerging Internet Technologies," *IEEE Transactions on Learning Technologies*, 2009.

[4] N. Suosa et. al, 'An Integrated Reusable Remote Laboratory to Complement Electronics Teaching', IEEE Transactions on learning technologies 2010

[5] M. A. Bochicchio et. al, "Hands-On Remote Labs: Collaborative Web Laboratories as a Case Study for IT Engineering Classes", *IEEE Transactions on Learning Technologies*, 2009

[6] M. Cooper et. al, "Remote Laboratories Extending Access to Science and Engineering Curricular", *IEEE Transactions on Learning Technologies*, 2009.

[7] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan "A Programmable and Configurable Mixed-Mode FPAA SOC," *IEEE Transactions on VLSI*, available on IEEE Xplore January 2016.

[8] M. Collins, J. Hasler, and S. George, "An Open-Source Toolset Enabling AnalogDigitalSoftware Codesign," *Journal of Low Power Electronics Applications*, January 2016.

[9] W. Wolf, "Hardware-software co-design of embedded systems," *Proceedings of the IEEE*, 1994, 967–989.

[10] 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. volume 7, pages 6:1–6:30, June 2014.

[11] C. R. Schlottmann and J. Hasler, High-Level Modeling of Analog Computational Elements for Signal Processing Applications *IEEE Trans on VLSI*, 2014.