Transforming Mixed-Signal Circuits Class through SoC FPAA IC, PCB, and Toolset

Jennifer Hasler, Sihwan Kim, Sahil Shah, Farhan Adil, Michelle Collins, Scott Koziol, and Stephen Nease,
Electrical and Computer Engineering (ECE)
Georgia Institute of Technology, Atlanta, GA 30332–250 USA E-mail:jennifer.hasler@ece.gatech.edu

***Abstract*—We present a SoC large-scale Field Programmable Analog Array (FPAA) test system, enabled by configurable analog–digital ICs to create a simple interface for a wide range of experiments in classroom environments. This configurable system appears as a simple digital peripheral using a standard USB interface for communication and power. Combined with a high-level tool flow for on-chip application design, this FPAA device demonstrates a number of mixed-signal computations, classification, and signal processing and its impact on hands-on circuit instruction.**

The paper presents the technical updates to our hands-on Integrated Circuit (IC) course, Neuromorphic Analog VLSI Circuits, ECE 6435. This course has been a first testbed for classroom innovation, including hands-on laboratory experiences as well as use of inverted classroom concepts (e.g. [1], [2], [3]). The primary focus is explaining the novel infrastructure, enabled by a new generation of large-scale Field Programmable Analog Arrays (FPAA), the SoC FPAA devices [4], developed at GT to eventually be available to the wider community. Our group is in the middle of assessment of using the SoC FPAA and tools for courses, and we look forward to report these results when completed.

The resulting SoC FPAA, and resulting infrastructure, creates a different student user experience. We want a simple and accessible laboratory framework, enabling a student laptop based setup using a SoC FPAA device and/or remote SoC FPAA device designed through graphical Scilab/Xcos based tools. Figure 1a shows a photo of one of multiple working FPAA board designs. The core technology is the introduction of the SoC FPAA devices [4], along with their programming infrastructure [5], and resulting system synthesis toolflow [6]. This approach gives a simple digital peripheral using a standard interface (i.e. USB), enabling a small Internet of Things (IoT) block. 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 all features on the resulting system.

The ECE6435 course[1] is built on experience with Caltech's Analog VLSI and Neural Systems course (CNS 182), framed in a single semester within an electrical engineering (as opposed to Computation and Neural Systems) department. This course was first offered in 1999, and has been continuously offered ever since. The original model of this course, similar to the CNS 182 structure, required those teaching the course to both fabricate Integrated Circuits (IC) ahead of the start of class, as well as providing the resources of many dedicated, computer-controlled, lab stations (scopes, ammeters, voltage supplies) for student use. With the start of using early FPAA devices in courses (F2005, F2006) [1], and the progression of new FPAA devices used in courses [2], [3], one eliminates the need for custom IC fabrication, as well as one sees a reduction of the required test hardware to a simple USB connected board into the student's laptop computer as demonstrated through this paper.

[1]The course information is openly available at the website: http: users.ece.gatech.edu/phasler/ECE6435.

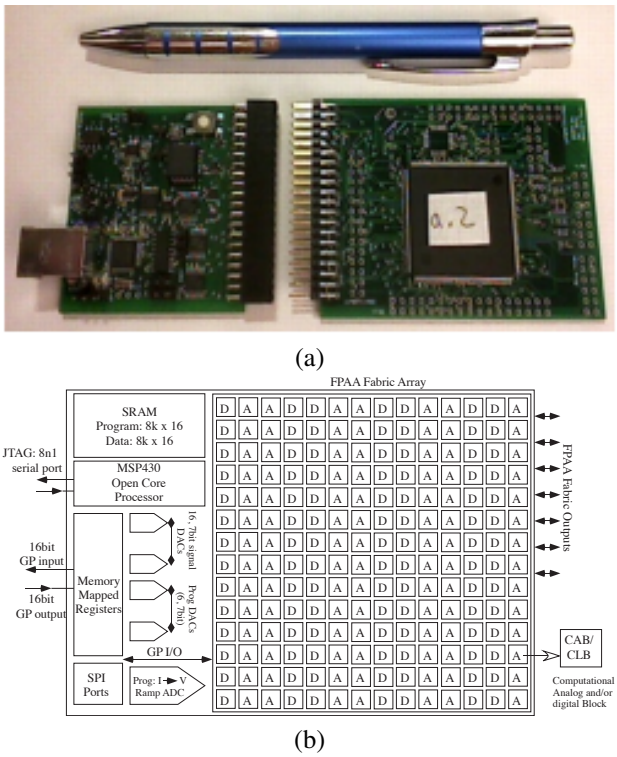

Fig. 1. SoC FPAA infrastructure for demonstration presentation. (a) Picture of the full SoC FPAA board, including two parts, a control board interfacing to the USB interface, and an IC board interfacing to the particular FPAA IC. (b) RASP 3.0 functional block diagram illustrating the resulting computational blocks and resulting routing architecture. The mixed array of the FPAA fabric is composed of interdigitated Analog (A) and Digital (D) configurable blocks on a single routing grid. The infrastructure control includes a μP developed from an open-source MSP 430 processor, as well as on-chip DACs, current-to-voltage conversion, and voltage measurement, to program each Floating-Gate (FG) device.

Fall 2006

Typical Topic Progression	Weeks
MOSFETs + FPAAs Intro	2
MOSFETs: Gain + Amplifiers	2
FG Intro, Program, switches	2
Program Current Sources	2
Differential Amps + OTAs	2
Integrate + Fire Neurons	2
Second-Order + Cochleas	2
Total	14
Final Exam, No final project	

Spring 2012

Typical Topic Progression	Weeks
Transistors	3
Amplifier Circuits (Int & Fire, WTA)	2
Transistor Channel Neurons	3
VMM + Basic Classifiers	2
Total	10

Spring 2016

Typical Topic Progression	Weeks
Introduction to FPAA	1
2 MOSFET circuits, OTAs	1
OTA Circuits / Filters	1
Developing Xcos model (macromodeling)	2
Transistor Channel Neurons	2
Analog Classification (VMM +WTA)	1
Dendrite Model + Compute	2
Total	10

Fig. 2. Snapshot comparison of curriculum of the project topics and time for each assignment. FPAA devices impact the types of material discussed, particularly enabling higher signal processing concepts in a given semester. Spring 2012 class (and later) all had final design projects measured on the FPAA device for the last 4-5 weeks of the course.

I. Recent FPAA course development

The SoC FPAA devices enable an innovative approach from previous laboratory classes. Even previous versions of this class [1] would have required significant bench infrastructure, and even earlier generations of this class would require full bench setups for students. These issues required significant scheduling issues for testing and measuring custom ICs.

The curricular development progressed along the last four times ECE 6435 was offered (Sp 2012, 2014, 2015, and 2016). From Spring 2012, where we primarily utilized earlier FPAA devices and initial toolset developed in MATLAB / Simulink, consistent with previous discussions [2], we began using the SoC FPAA architectures and tools, as we worked through some of the tool and hardware issues. Sp2012 also started incorporating design and hardware verification given the flexibility of the FPAA platform.

Spring 2015 enabled full integration of our current tools (Section III) into the class. The lessons learned have made this Spring 2016 class operate fairly smoothly with this new infrastructure. The students used the FPAA tools and hardware right from the beginning of class, where students used the tools and hardware for measurements during the second class meeting. These classes benefited from an inverted classroom format, with multiple taped lectures opening time in class for experimental measurements, enabling student projects, typically in groups of 2, focused on design and experimental measurements verifying that design.

Starting in Spring 2015, we introduced using a remote SoC FPAA test setup [7]. Using the remote board is fairly similar to using a board in hand, except that the user selects another GUI button that emails the files to a location the remote device can access, sending a return email when the system is executed on that device. These techniques are ready to implement in other ECE courses, initially analog and mixed signal courses, using design projects throughout the class.

Figure 2 shows a comparison between the material coverage from three different years. The 2006 class, the first year heavily using the FPAA device, looks considerably different from the course in 2016 due to the evolution of the technology. This year required considerable time and effort on FG devices, programming, and low-level infrastructure, concepts that are all abstracted in high level tools, for the most part. Further, the 2006 class barely gets to the first 3-4 weeks of the 2016 class, without opportunity for doing a final design project (was a final exam that year). In recent years (2014-2016), the focus moved away from a few multi week projects, due to lack of consistent student focus. Weekly projects helps build and keep the necessary scaffolding in place. In 2016 class, the two week labs correspond to the window when other classes are typically scheduling exams.

II. Overview of the SoC FPAA IC

Figure 1b shows the SoC FPAA device [4], an analog–digital programmable and configurable IC system. This configurable fabric effectively integrates analog (A) and digital (D) components in a hardware platform easily mapped towards compiler tools. The switchable analog and digital devices are a combination of the components in the Computational Analog Blocks (CAB), in the Computational Logic Blocks (CLB), and the devices in the routing fabric that are programmed to non-binary levels. The design of the routing fabric was not a block of analog components and a block of digital components with hard-build data converters in between, but rather a mixed fabric to explicitly allow the lines to be blurred as the application requires. The interaction of analog computation, digital FPGA-like components, and a μP (open-source MSP 430 microprocessor) infrastructure coming together creates a significant co-design space between these three domains (analog, digital, μP). This FPAA further employs on-chip structures for 7bit signal DACs, a ramp ADC, and memory-mapped General Purpose (GP) IO.

This SoC FPAA enables nonvolatile digital and analog programmability through Floating-Gate (FG) devices, both in the routing fabric, but also for parameters for the computing elements. These devices enable useful computation out of using routing resources, such as Vector-Matrix Multiplication (VMM) built out of the routing crossbar (i.e. floating-gate) switch matrices. A typical user will often use digital (GPIO) or

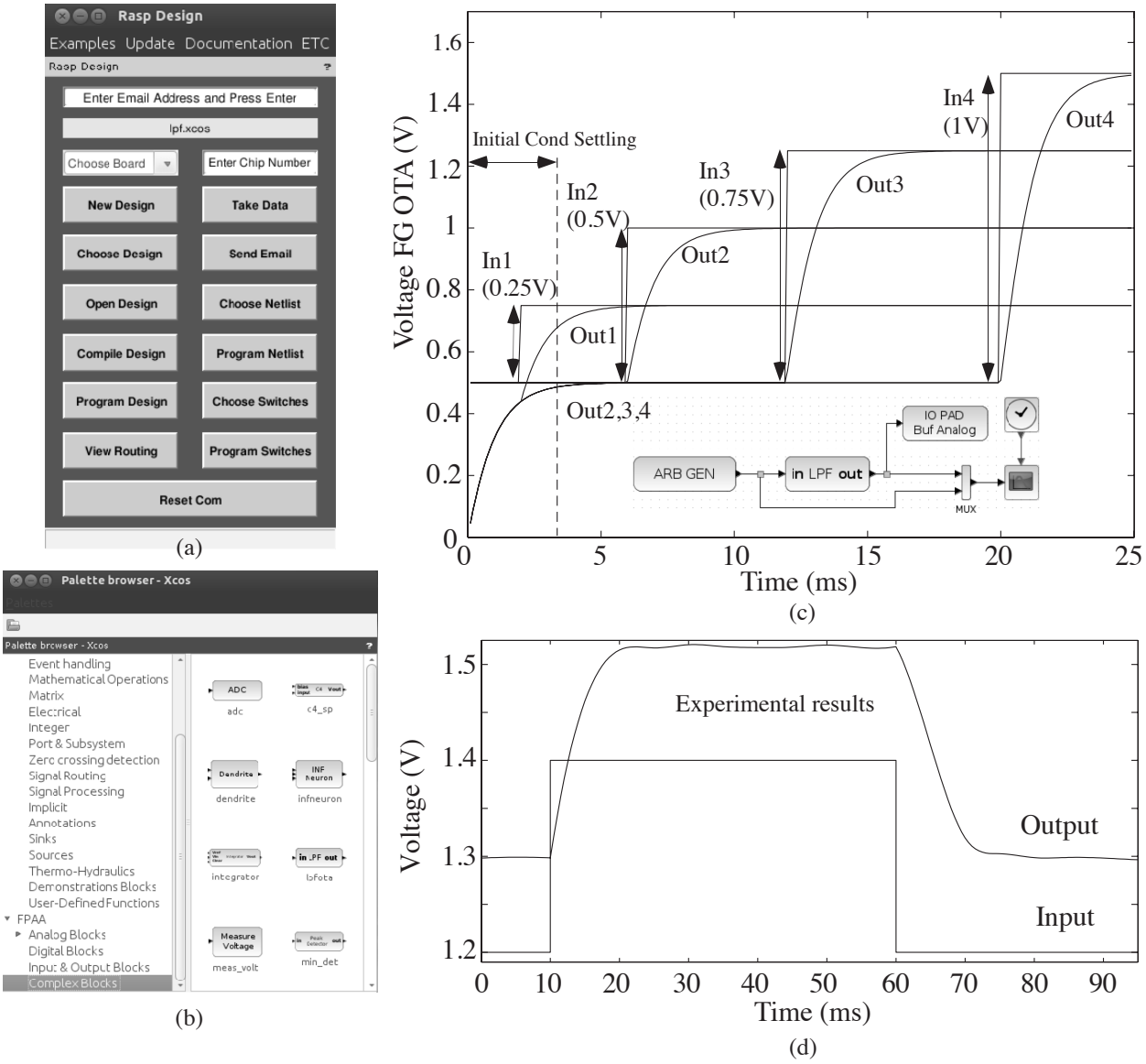

Fig. 3. An example of the entire tool flow for a Low-Pass Filter (LPF) computation. (a) The user chooses basic design options through the FPAA Tools GUI, which starts running when the Scilab tools are started in the distributed Ubuntu Virtual Machine (VM). (b) Snapshot of the Xcos palette for FPAA blocks. There are four sections, namely the Analog, Digital, Input/Output and Complex Blocks; the Analog, Digital and I/O blocks consist of basic elements in different tiles of a chip. Complex blocks are pre-defined circuit blocks made of more than one basic element. (c) Simulation results for 4 input and output computation. Lines, and resulting blocks, allow for vectorized as well as scalar inputs. Inset shows the Xcos diagram; the user sets parameters for simulation or for compiling into IC. (d) Experimental results for a 1 input and output computation.

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 (e.g. 8bits) through GPIO or voltage measurement through slower 14bit ADC. FG programming enables on-chip programming requiring simply a downloaded file into the processor, and taking no SRAM memory for the FPAA's normal operation [5]. Typical measured FG accuracy is tighter than 1 percent for subthreshold currents, relating to less than 250μV variation.

III. User Tool Framework Summary

This FPAA SoC practically requires a toolset for system design in a reasonable design timeframe. A user of these ICs can *compile* their design, rather than undertaking a full custom IC design process, typical of current FPGA devices. Compilation means the user starts with a description of the IC, going from a range of high-level block abstractions to a literal list of FG switches (and intermediate representations like verilog), to an experimentally measured IC utilized in the same places as a commercial IC.

Higher-level tools also allow the use of these systems in educational experiences [1], [2], which will be essential to educating engineers to *design* for system applications. This open-source tool set explicitly enables a wider user community for mixed-signal configurable designs. The tool integrates a high-level design environment built in Scilab and Xcos (an

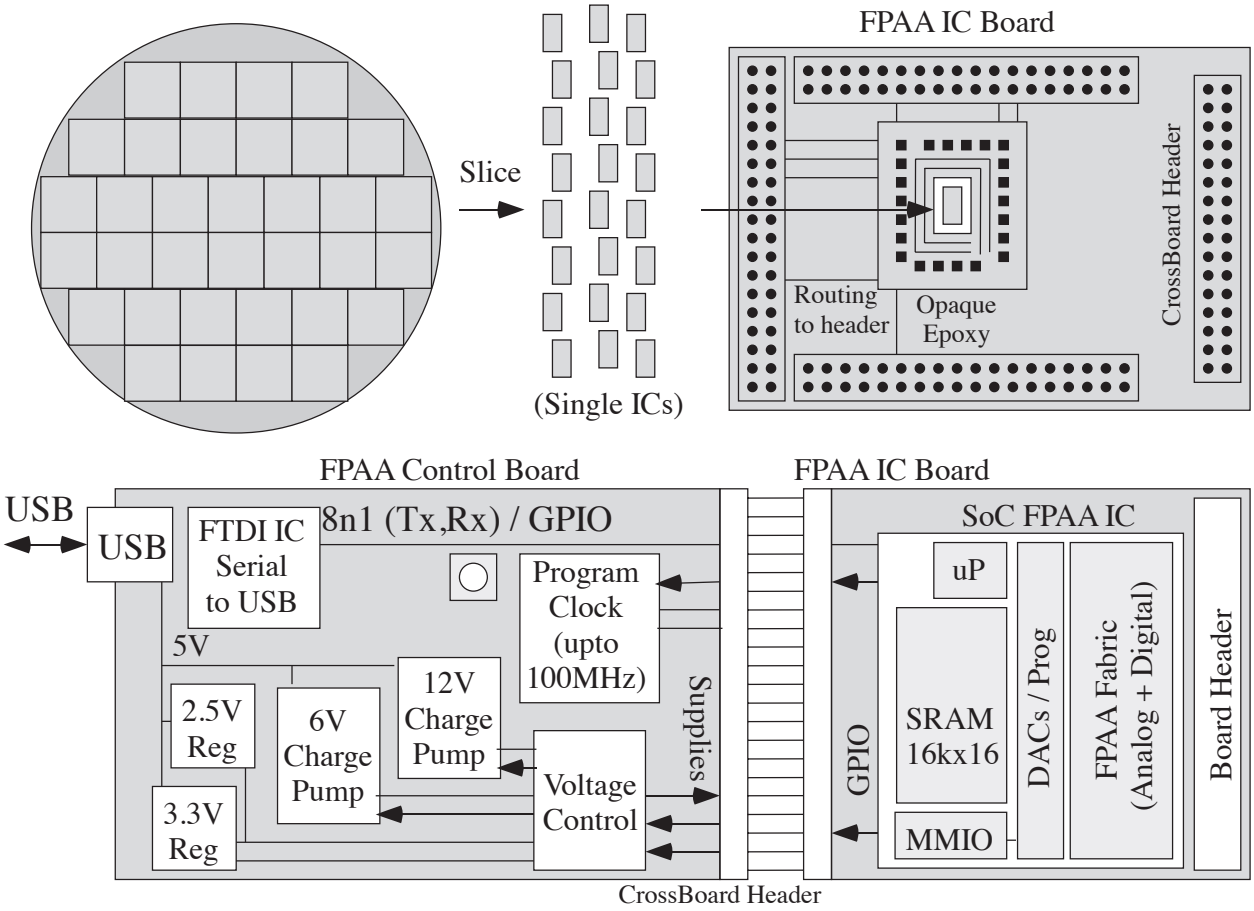

Fig. 4. Detailed Picture of the SoC FPAA board. **Upper Figure**: Process flow for Chip-on-Board (CoB) build for the resulting IC board. **Lower Figure**: Detailed block diagram for the SoC FPAA board. The FPAA control board primarily handles the USB to serial communication interface, the programmable clock generator, as well as the multiple supply voltages, controlled by the FPAA, required for operation and programming of the FPAA device.

open-source clone for MATLAB and Simulink, respectively), with a compilation tool, **x2c**, to compile from high level description to a targetable switch list to be programmed on the FPAA device. An example of our tool is shown in Fig. 3. **x2c** is composed of *sci2blif*, translating a block in scilab to .blif format. Our modified VPR tool [8] takes .blif and verilog formats, and compiles to hardware descriptions through our code **vpr2swcs**. The chip details are specified in architecture files for analog–digital SoC. The graphical high level tool uses a palette for available blocks that compile down to a combination of digital and analog hardware blocks, as well as software blocks on the resulting processor.

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 now 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.

IV. SoC FPAA board

Figure 4 shows a detailed diagram of the SoC FPAA board. The IC and interface elements is a single system, acting as a peripheral through a USB port, appearing to be a standard peripheral to a typical device. A single USB port results in a self-contained programmable and configurable platform. The programmable interface through the high-level tools connects the analog and digital capability of the FPAA IC, the resulting input and output blocks for the system, as well as the resulting system control. The entire IC is the computing system; all components are part of the computation.

We openly distribute this tool set as an open-source Virtual Machine (VM)[2], as well as associated PC board files [3]. Figure 5 shows the user-level picture of the SoC FPAA

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

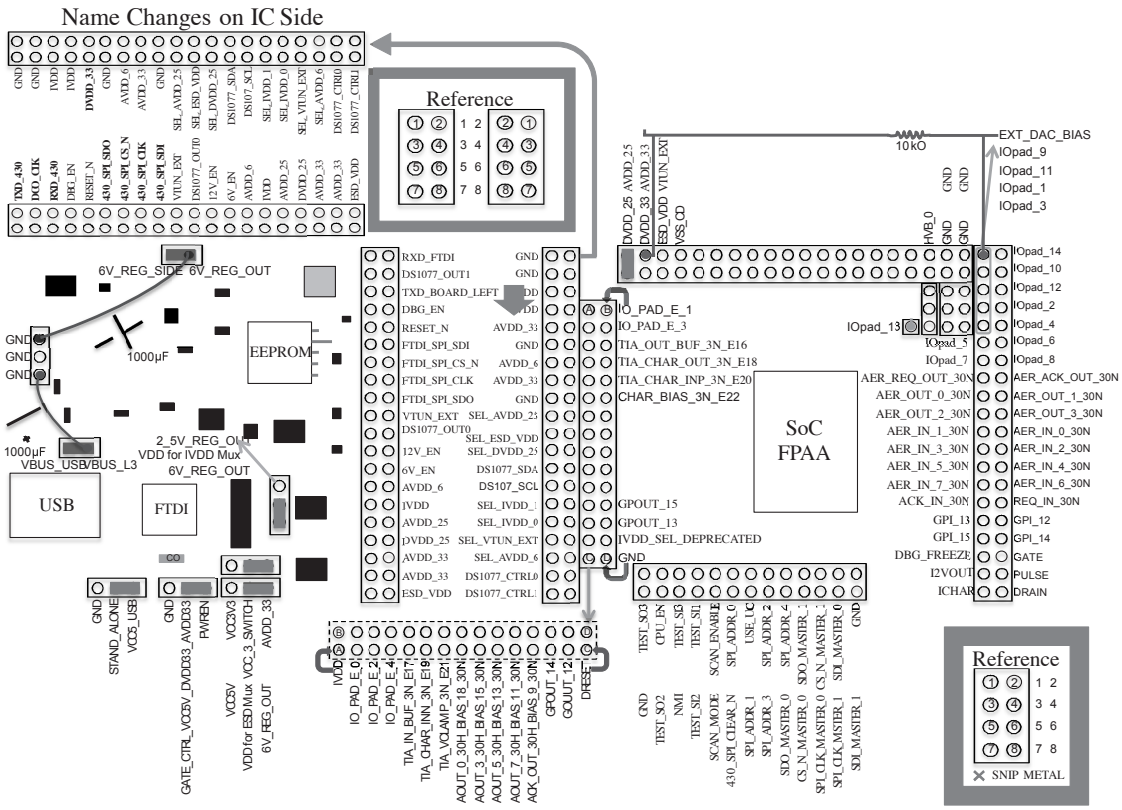

Fig. 5. User level view of the SoC FPAA IC board (both control and IC board). For students taking this class, this would be their perspective of the FPAA board, although most of the jumper pins are abstracted away for ease of use.

board. This VM and reference board design encourages a user community (classroom use, research groups, as well as interested users) around these tools, enabling this community to further improving this tool set. The toolset is encapsulated into an Ubuntu 12.04 VM, simply requiring one button to bring up the entire graphical working toolset. The primary student hardware or software challenge involves the occasional USB communication errors through the VM, through the host OS, through the device.

Figure 4 shows the block diagram of the control board part of the FPAA board. The primary off-chip infrastructure is μP IC controlled high-voltage power handling (12V and 6V charge pump ICs); these components were left off chip to minimize the IC design risk but require additional board level infrastructure. A USB to serial converter IC was chosen to interface to the μP. 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 4 shows the block diagram of the FPAA on the IC board; the FPAA system technology file specifies the (user chosen) configurability.

V. Example FPAA Measurements

We will show some representative measurements for our SoC FPAA system. Several measurements utilize FG transistors in a CAB, processor, signal DACs and memory mapped register in a single application. These techniques blur the line between device under test and the entire operating system.

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

Figure 6 an example using the FPAA tools with either the FPAA board, or the remote FPAA board. Figure 6 shows measured data for a parallel bank of bandpass filters and amplitude detection typically used for low-power sub band analysis. This design illustrates the co-design between assembly, analog, digital components, and interfacing. The corner frequencies are programmed between 100Hz and 750Hz, seen by the different peaks in the chirp response in Fig. 6. This example uses the same SRAM memory, 7bit input DAC, and compiled ADC (4KSPS). A similar chirp signal linearly varies

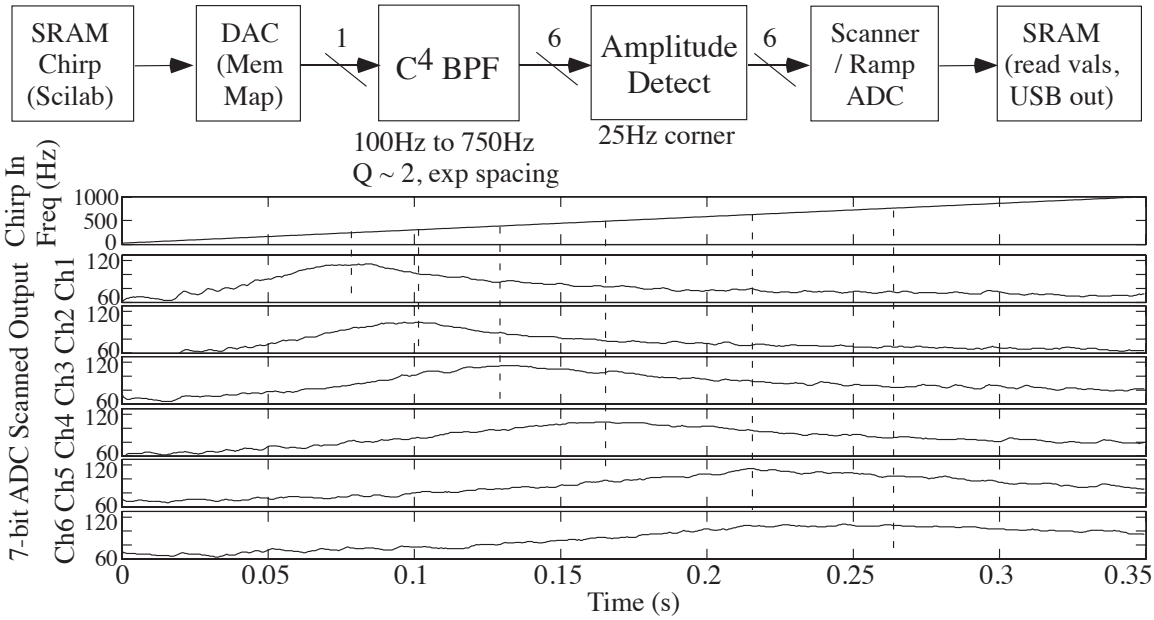

Fig. 6. Block diagram (similar to Scilab / Xcos definition) and output results for a bank of 6 BandPass Filter (BPF) and Amplitude detect elements compiled and measured through a remote FPAA board. The input chirp signal, a linear sweep between 25Hz and 1kHz, is stored on-chip SRAM to be played through a memory-mapped DAC. The output signals went through a demultiplexing module, compiled using the shift-registers located in routing fabric, and then through the same 8-bit ramp ADC module; this measurement only used the upper half of the positive values (effectively 6bit). The linear frequency sweep with time is illustrated, as well as the resulting outputs of all 6 frequency channels, each programmed to a different frequency location (exponentially spaced).

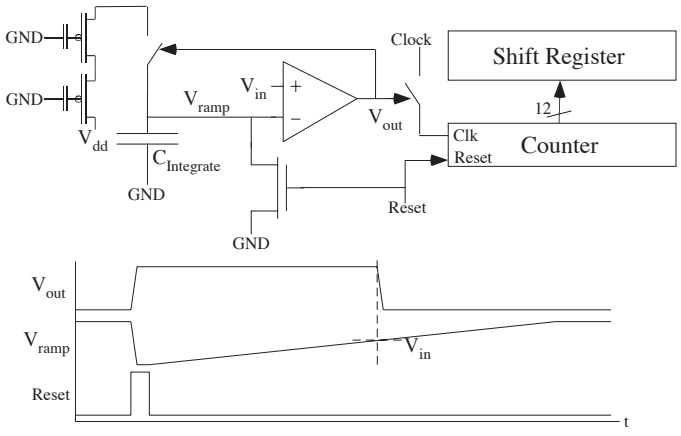

Fig. 7. One key aspect is the range of potential methods for voltage measurement in the fabric to connect to the μP device. Typically, these devices include direct digital inputs, with resulting level comparisons, Analog-to-Digital Converters (ADC), as well as a range of other classifier components. One can compile a ramp ADC in the routing fabric to operate at a range of sample frequencies and resolution.

between 25 and 1kHz.

Figure 7 shows circuit compilation at the analog–digital boundary through compilation of a ramp ADC, illustrating the flow from input SRAM memory data to the computed output SRAM memory data. The resulting simple 8bit ramp ADC uses the digital infrastructure and μP to move analog signals into stored SRAM memory. This structure 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.

REFERENCES

[1] C. Twigg and P. Hasler, "Incorporating Large-Scale FPAAs Into Analog Design and Test Courses," *IEEE Transactions on Education*, Vol. 51, No. 3, 2008, pp. 319-324.

[2] P. Hasler, C. Schlottmann, S. Koziol, S. Ramakrishnan, S. Brink, and A. Basu, "FPAA chips and tools as the center of an Design-Based Analog Systems Education," *IEEE MSE*, San Deigo, June 2011, pp. 47-51.

[3] M. Collins, J. Hasler, and S. George, "Analog systems education: An integrated toolset and FPAA SoC boards," *IEEE MSE*, Pittsburg, May 2015, pp. 3235.

[4] 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*, February 2016.

[5] S. Kim, J. Hasler, and S. George, "Integrated Floating-Gate Programming Environment for System-Level Ics," *IEEE Transactions on VLSI*, February 2016.

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

[7] S. Shah, J. Hasler, S. Kim, I. Lal, M. Kagle, and M. Collins, "Demonstration of a Remote FPAA System for Research and Education," *IEEE ISCAS*, May 2016.

[8] 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.

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

[10] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, S. Ramakrishnan, P. Hasler "Hardware and Software Infrastructure for a Family of Floating-Gate FPAAs," *IEEE IEEE International Symposium on Circuits and Systems*, June 2010, pp. 2794-2797. Winner of the best demonstration paper award.

[11] 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," *ACM TRETS*, Vol. 7, No. 2, June 2014, pp. 6:1 - 6:30.

[12] Scilab Enterprises. *Scilab: Free and Open Source software for numerical computation*. Scilab Enterprises, Orsay, France, 2012.