Hardware and Software Infrastructure for a family of Floating-Gate Based FPAAs

Scott Koziol, Craig Schlottmann, Arindam Basu, Stephen Brink, Csaba Petre, Brian Degnan,
Shubha Ramakrishnan, Paul Hasler, and Aurele Balavoine
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332–0250
Email:{skoziol,phasler}@ece.gatech.edu

***Abstract*—Analog circuits and systems research and education can benefit from the flexibility provided by large-scale Field Programmable Analog Arrays (FPAAs). This paper presents the hardware and software infrastructure supporting the use of a family of floating-gate based FPAAs being developed at Georgia Tech. This infrastructure is compact and portable and provides the user with a comprehensive set of tools for custom analog circuit design and implementation. The infrastructure includes the FPAA IC, discrete ADC, DAC and amplifier ICs, a 32-Bit ARM based microcontroller for interfacing the FPAA with the user's computer, and Matlab and targeting software. The FPAA hardware communicates with Matlab over a USB connection. The USB connection also provides the hardware's power. The software tools include three major systems: a Matlab Simulink FPAA program, a SPICE to FPAA compiler called GRASPER, and a visualization tool called RAT. The hardware consists of two custom PCB designs which include a main board used to program and control an FPAA IC and an FPAA IC adaptor board used to interface a QFP packaged FPAA IC with the 100 pin ZIF socket on the main programming and control board.**

***Index Terms*—FPAA, Simulink, Reconfigurable Analog**

Field Programmable Analog Arrays (FPAAs) are useful for both research and teaching [1]. Previous Georgia Tech FPAA ICs have been programmed using a self contained development platform that included a commercial FPGA development board, a custom FPAA board, and a AC-DC power module. This previous hardware platform [2] fit into an enclosure about the size of a shoe box and communicated with the computer using ethernet. The new platform described in this paper is significantly smaller (about twenty-six square inches) and communicates over USB. The power supply systems have also been changed and improve upon the portability. A picture of the programming and control board is found in Fig. 1, and a block diagram of the system is found in Fig. 2. The infrastructure is a proven system as it has seen use in a DARPA project workshop at the University of Southern California; two workshops in Telluride, CO; and two more workshops at Gerogia Tech. The boards have also been the primary laboratory infrastructure for two semesters of a graduate course in Neuromorphic Analog VLSI Circuits at Georgia Tech.

Section I discusses the basics of FPAAs, Section II discusses software tools, Section III discusses the hardware, Section IV shows the design flow using a demonstration test circuit, and Section V is a closing summary

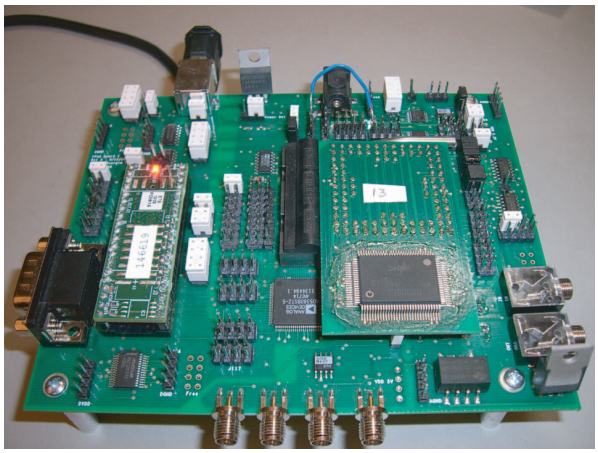

Fig. 1: FPAA Programming & and Control Board (25.76 square inches). Note the USB connection on the top left; 40 pin DIP microcontroller module to the right of the serial connector; the 100 pin ZIF socket for inserting the FPAA ICs; many 2x4 pin headers connected to FPAA I/O, DAC outputs, ADC inputs, FPAA control pins, and power/ground; 4 SMA for FPAA I/O interface; and the audio jacks (on the lower right).

I. Implementating circuits on a field-programmable analog array (FPAA)

A family of floating-gate based large-scale FPAAs are being developed at Georgia Tech. This family includes the Reconfigurable Analog Signal Processor (RASP) IC [3], [4], the Reconfigurable Smart Sensory Chip (RSSC), [5] as well as FPAAs with biological and adaptive computational blocks. In this work we present the latest software and hardware used to map circuits onto these FPAAs.

The RASP and RSSCs are reconfigurable analog platforms that utilize a switch matrix of programmable floating gate transistors as switch elements. The reconfigurable nature of the platforms allow rapid building and testing of different circuit configurations [4].

Fig. 3 shows a RASP 2.8 block diagram and resulting die photo of the working IC. The IC is arranged in an 8 x 4 array of computational analog blocks (CAB)s, with two CAB versions: one that includes a 4x4 vector-matrix multiplication (VMM) placed along the top and bottom rows, and one that does not. The arrays have a mixture of analog granularity, so that one has access to transistor-level functions, as well as some higher signal processing features. Programmable floating-gate circuit technology enables the FPAAs to provide area-efficient,

978-1-4244-5309-2/10/$26.00 ©2010 IEEE

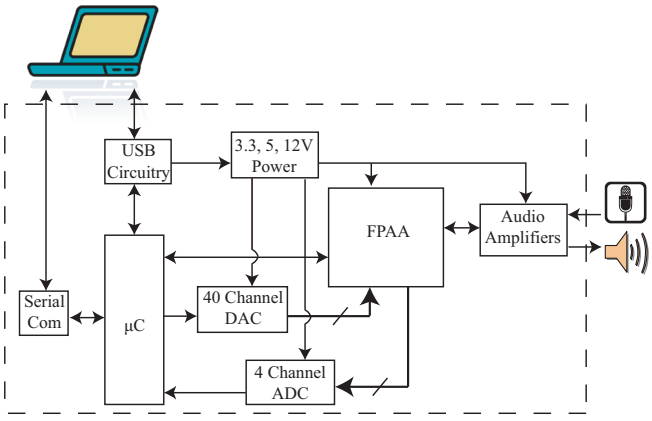

Fig. 2: Block Diagram of the FPAA programming and control board. The board has been designed to be self contained and portable, only needing a laptop. The user chooses between USB or serial communication. The power is supplied by the USB port. The microcontroller (μC) is a 40 pin DIP plug-in module which uses an ATMEL 32 Bit ARM processor. The FPAA I/O can be reconfigurably connected to the discrete ADC and DACs using headers and jumpers. MP3 players can easily be used as inputs to the FPAA by using the audio input port and audio amplifiers.

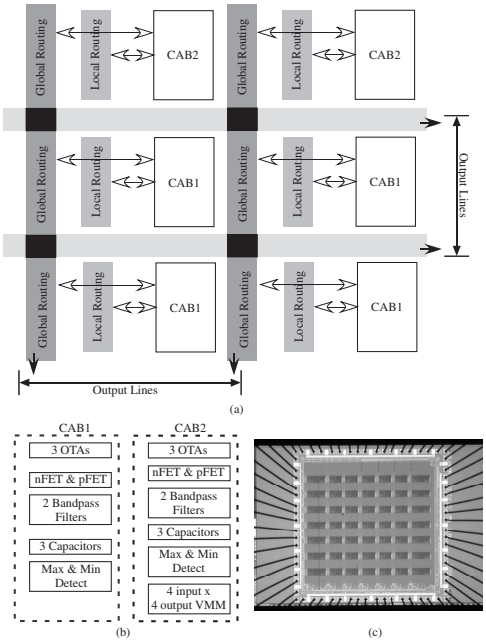

Fig. 3: A Reconfigurable Analog Signal Processing IC. (a) Block Diagram of the RASP 2.8 IC; the IC utilizes 32 CABs in a multi-level routing scheme. (b) Block diagram of the CAB components. (c) Die Photo of the RASP IC which consumes 3mm x 3mm area in 0.35μm CMOS process.

accurately programmable analog circuitry that can be easily integrated into a larger digital/mixed-signal system [3] [2]. The programmable elements allows for switch elements that have a dual role as computational elements [6].

A closed loop control system is used to program the floating-gate elements [7]. Using the software tools that will be described in Section II, a list of FPAA switches is first selected to be programmed. Some floating-gate switches may be *fully on* whereas others may be programmed to specific currents. Matlab and the ARM Core Microprocessor are the brains of the control system. The Matlab interface needs to communicate to the FPAA that it would like to program switch *X* to some current *Y*. It does this by sending a message via USB to the FPAA board's ARM core microprocessor. This microprocessor then communicates switch coordinates and calculated programming voltages to the FPAA IC using an SPI bus. This completes the active programming part of this iteration. The FPAA IC's on-chip programming structures then measure the result of this programming iteration and reports the level to which the floating-gate is programmed. This feedback is communicated from the FPAA IC to the ARM core microprocessor using SPI. In some cases the microprocessor uses this feedback to control the FPAA IC, in other cases the feedback is communicated from the microprocessor to MATLAB over USB and corrective signals are calculated in MATLAB. This iterative control loop continues until the desired floating-gate switch current is reached.

II. FPAA Software Infrastructure

The FPAA software tools include three major systems: a Matlab Simulink FPAA program, a SPICE to FPAA compiler called GRASPER (Generic Reconfigurable Array Specification & Programming Environment), and a visualization tool called RAT (Routing Analysis Tool).

The Matlab Simulink Tool is an automation tool which converts Simulink models to a Spice netlist, which can then be automatically compiled to FPAA targeting code and implemented on an FPAA. This allows DSP and neuromorphic engineers to have a fast method of implementing low power analog solutions without having to gain the necessary expertise in circuit design [8] [9]. The current system allows the user to program floating gates at a rate of approximately one per second. This Simulink Tool is described in greater detail in [10].

The GRASPER tool converts a circuit's SPICE file into a list of FPAA switches that implement the circuit on the FPAA. GRASPER has features which grant the user various levels of control as to what components are used and how the circuit is mapped onto the FPAA IC [11] [12] [13] [14].

The RAT is a Matlab GUI which graphically shows the topology of how a circuit is routed on the FPAA switches [15]. New designs can be created or existing designs can be modified by pointing and clicking with the mouse. The RAT can easily be configured to support FPAA ICs with different number of computational analog blocks.

Once an FPAA is targeted, the user can interface with the FPAA I/O in a couple of ways. First the user can jumper FPAA I/O to the discrete DAC and/or ADC ICs. These ICs are controlled through the Matlab interface. Table I lists a few sample Matlab commands. The user can also interface with the FPAA by using the audio amplifier ports. Fig. 5 shows the legend used to identify the various headers. There are thirty-two header pins on the board that are considered "factory settings" and are normally jumpered to specific control pins.

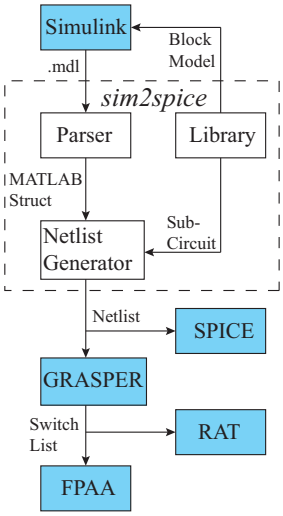

Fig. 4: Software flow for designing systems on the FPAA. Top level designs are done in Simulink. *Sim2Spice* converts it to a Spice netlist, which can then be compiled into an FPAA switch list. [10]

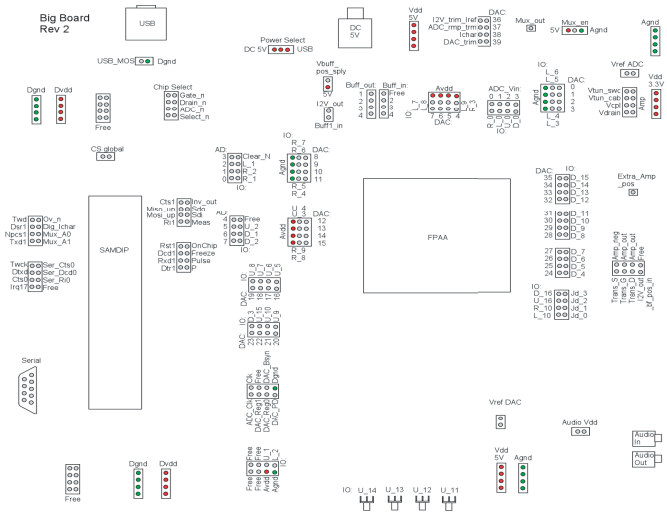

Fig. 5: Header Map used as a legend to identify pins on the programming and control FPAA board. U_*, D_*, L_*, R_* are FPAA I/O pins

III. FPAA Hardware Infrastructure

The hardware consists of two custom PCB designs: a main programming and control board and an FPAA IC adaptor board. The programming and control board, Fig. 1, is the workhorse of our FPAA infrastructure family. It has header pins which allow easy access to most of the FPAA pins. This is helpful for many things including power measurements, circuit debugging, etc. This board has a 100 pin zero insertion force (ZIF) socket into which the FPAA IC is placed. This socket makes this board a good general platform for testing many families of FPAAs such as our General, Sensor, Bio, MITE, and Adaptive versions. Our FPAAs are typically packaged in plastic surface mount packages. We have developed an adaptor board PCB, Fig. 6, which connects the surface mount packages to pins. These pins then plug into the ZIF socket on the

TABLE I: Sampling of Matlab commands used to interface with FPAA

Matlab Function	Description
SET_DAC_USB	sets one of the 40 channels on the DAC IC
READ_ADC_USB	reads into Matlab a value from the ADC IC.
PROGRAM_aa	used to program a list of elements on the FPAA

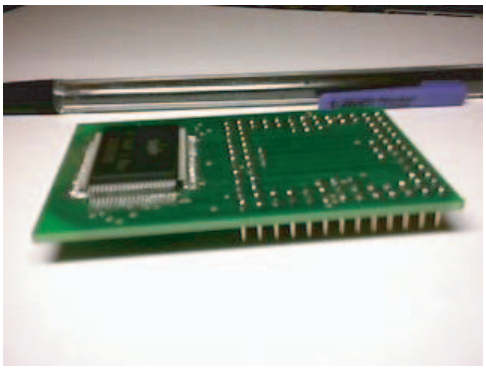

Fig. 6: Adaptor Board (4.16 square inches) This custom PCB has a quad flat pack (QFP) packaged FPAA IC on one side and pins on the other side. The pins plug into the 100 pin ZIF socket on the programming and control board.

programming and control board.

The programming and control board has the following features: USB or Serial communication capabilities, USB power or external DC power, SMA connectors for connecting to FPAA I/O pins, a discrete 14-Bit DAC IC which has forty channels (most of which can be used as inputs to FPAA I/O pins), a discrete 8-bit ADC that can also be used to connect to FPAA I/O pins, amplifiers to be used as I/O buffers, and finally an audio amplifier and audio jacks which can be used for audio input and output connections to the FPAA. The board also has 3.3V, 5V, and 12V supplies. The board uses an Atmel AT91SAM7S ARM based microcontroller to communicate via USB to the any desktop or laptop computer. The software emulates a serial communications device class (CDC) connection, and most modern operating systems have drivers for this software out-of-the-box. The ARM Core microprocessor on the board was purchased as a 40 pin DIP plug-in module.

IV. Design Flow using a Demonstration Test Circuit

Fig. 7 shows the design flow for implementing a lowpass filter on an FPAA. First a Simulink block diagram of the system was generated, Fig. 7(a). Although not shown in this picture, this system can be digitally simulated in Matlab. Next, the *Sim2spice* tool, Fig. 4, is used to generate a SPICE file from the Simulink block diagram, Fig. 7(b). Fig. 7(c) shows the text file which is the output of the GRASPER compiler. The first two numbers in each line are the row and column locations of a particular floating-gate transistor on the FPAA, and the last number on the line represents the desired current in the transistor. Fig. 7(d) shows the topology of the GRASPER routing on a RASP 2.9 IC. The switch list was targeted onto a RASP IC and a step input (blue) was applied to the input pin. The result was measured and shown in black in Fig. 7(e).

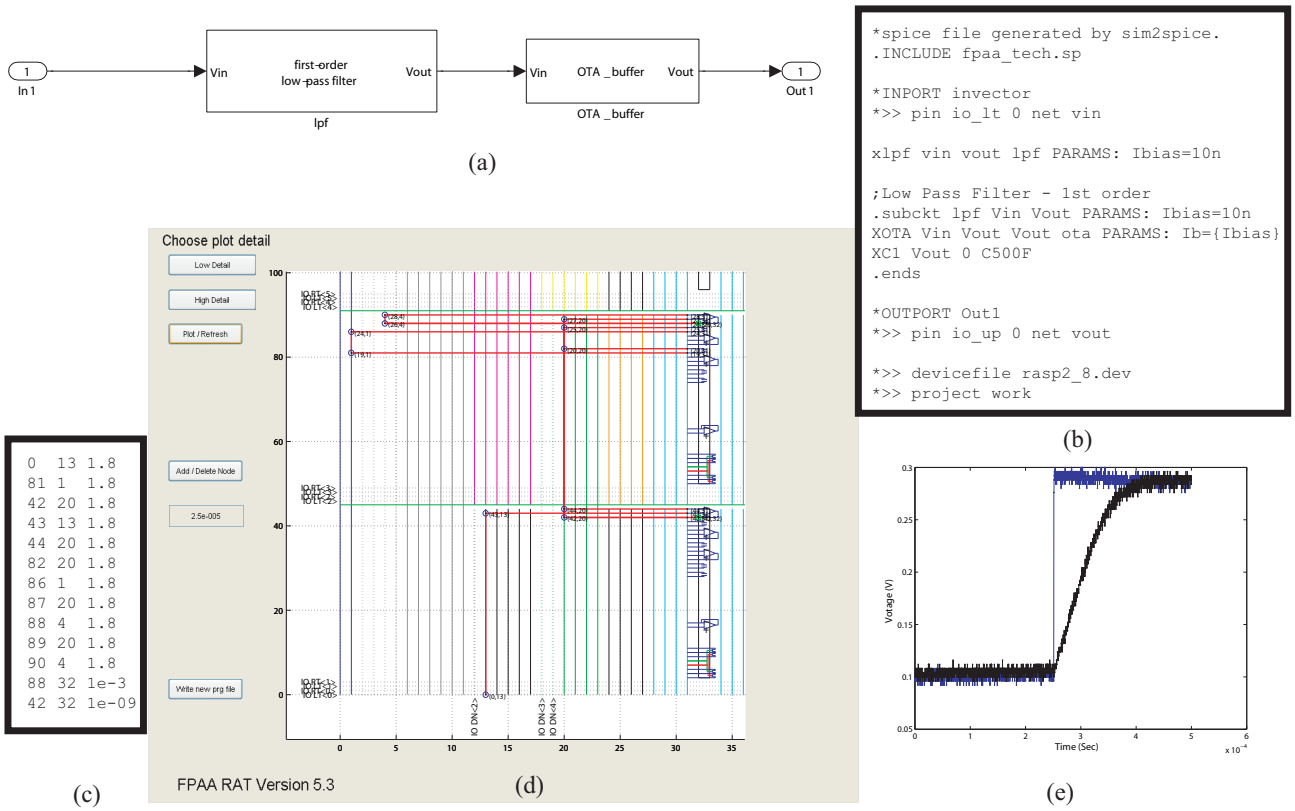

Fig. 7: Design Flow for a low pass filter (a) Simulink Block Diagram (b) SPICE list generated by *Sim2Spice* tool (c) FPAA switch list generated by GRASPER tool (d) RAT Figure showing switch list routing on RASP 2.9 IC (e) Measured Results from RASP IC, Blue is the input signal, black is the lowpass filterd output

V. Conclusions

We have described a comprehensive set of software and hardware tools that let users quickly and easily create custom AVLSI circuits. The software tools allow users with varying levels of circuit design experience to be successful at synthesizing circiuts. The hardware infrastructure platform allows users extreme flexibility to monitor and control the FPAA pins. The adaptor board allows users to quickly interchange FPAAs on the main programming and control board.

References

[1] C. Twigg and P. Hasler, "Incorporating large-scale fpaas into analog design and test courses," *Education, IEEE Transactions on*, vol. 51, no. 3, pp. 319–324, Aug. 2008.

[2] C. Twigg, P. Hasler, and F. Baskaya, "A self-contained large-scale fpaa development platform," in *Circuits and Systems, 2007. IEEE International Symposium on*, May 2007, pp. 1187–1191.

[3] T. Hall, C. Twigg, J. Gray, P. Hasler, and D. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 11, pp. 2298–2307, Nov. 2005.

[4] A. Basu, C. Twigg, S. Brink, P. Hasler, C. Petre, S. Ramakrishnan, S. Koziol, and C. Schlottmann, "Rasp 2.8: A new generation of floating-gate based field programmable analog array," in *Custom Integrated Circuits Conference, 2008. IEEE*, Sept. 2008, pp. 213–216.

[5] S.-Y. Peng, G. Gurun, C. Twigg, M. Qureshi, A. Basu, S. Brink, P. Hasler, and F. Degertekin, "A large-scale reconfigurable smart sensory chip," in *Circuits and Systems, 2009. IEEE International Symposium on*, May 2009, pp. 2145–2148.

[6] C. Twigg, J. Gray, and P. Hasler, "Programmable floating gate fpaa switches are not dead weight," in *Circuits and Systems, 2007. IEEE International Symposium on*, May 2007, pp. 169–172.

[7] A. Basu and P. Hasler, "A fully integrated architecture for fast programming of floating gates," in *Circuits and Systems, 2007. IEEE International Symposium on*, May 2007, pp. 957–960.

[8] C. Twigg, P. Hasler, and D. Anderson, "Large-scale fpaa devices for signal processing applications," in *Acoustics, Speech and Signal Processing, 2007. IEEE International Conference on*, vol. 2, April 2007, pp. II–69–II–72.

[9] C. Schlottmann, C. Petre, and P. Hasler, "Vector matrix multiplier on field programmable analog array," in *Acoustics, Speech and Signal Processing, 2010. IEEE International Conference on*, March 2010.

[10] C. Petre, C. Schlottmann, and P. Hasler, "Automated conversion of simulink designs to analog hardware on an fpaa," in *Circuits and Systems. IEEE International Symposium on*, May 2008, pp. 500–503.

[11] F. Baskaya, B. Gestner, C. Twigg, S. K. Lim, D. Anderson, and P. Hasler, "Rapid prototyping of large-scale analog circuits with field programmable analog array," in *Field-Programmable Custom Computing Machines. 15th Annual IEEE Symposium on*, April 2007, pp. 319–320.

[12] F. Baskaya, D. Anderson, and S. K. Lim, "Net-sensitivity-based optimization of large-scale field-programmable analog array (fpaa) placement and routing," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 7, pp. 565–569, July 2009.

[13] I. Baskaya, "Physical design automation for large scale field programmable analog arrays," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, Aug. 2009.

[14] F. Baskaya, D. Anderson, P. Hasler, and S. K. Lim, "A generic reconfigurable array specification and programming environment (grasper)," in *Circuit Theory and Design, 2009. European Conference on*, Aug. 2009, pp. 619–622.

[15] D. Abramson, "A mite based translinear fpaa and its practical implementation," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, Nov. 2008.