A Floating-Gate-Based Field-Programmable Array of Analog and Digital Devices

Richard B. Wunderlich, Farhan Adil, and Paul Hasler

***Abstract*—A hybrid, mixed-signal, reconfigurable system, based on the combination of a field-programmable analog array (FPAA) and FPGA is presented. The chip is a fine-grained interleaving of interchangeable analog and digital tiles, wherein a tile comprises either analog or digital computational elements and interconnect. The chip comprises the array core, high-speed I/O, and floating-gate programming infrastructure that communicates over a standard SPI bus. The FPAADD consists of 27 x 8 array of 108 digital and 108 analog tiles and peripheral circuitry on 5 x 5 mm^2 die fabricated in a 0.35-μm CMOS process. The chip's 132,000 FG switch elements are tunable from less than 10 $k\Omega$ to greater than 10 $G\Omega$ effective resistance at programming rates exceeding 1 μs for 1-b accuracy and 50 ms for 9-b accuracy. The computational analog blocks contain a variety of subcircuits: programmable offset wide-linear-range OTAs to nMOS and pMOS transistors, and interconnect. The digital tiles contain combinational logic blocks of homogenous clusterings of look-up tables and flip-flops and interconnect. The routing scheme was designed to reduce local interconnect parasitic capacitance and utilize a lightweight Manhattan style global interconnect. Measured results of low level computational elements and routing infrastructure and preliminary system results are presented.**

***Index Terms*—Analog signal processing, field-programmable gate array (FPGA), field-programmable analog array (FPAA), floating-gate (FG), reconfigurable system, mixed-signal system.**

I. Introduction

FPGAs are very popular, and are well proven in their ability to rapidly prototype, emulate, or implement digital systems. FPAAs, while no where near as mature or ubiquitous as FPGAs, are gaining in popularity, and are quickly proving themselves indispensable in the realm of analog prototyping, implementation, and even education [1], [2], [3], [4], [5]. So while systems exist for prototyping digital systems and systems exist for prototyping analog systems, there is little in the way of prototyping mixed-signal systems. In [6] they create a mixed-signal reconfigurable environment from discrete FPAAs, FPGAs, and data converters for synthesizing and prototyping mixed-signal systems. There has been some work to integrate this type of design into into SoCs [7], [8].

All of these designs, however, are essentially an FPAA array and an FPGA array isolated by dedicated data converters. The data converters are largely inflexible, and once fabricated, there is no way to change the percentage of area dedicated to analog versus digital. We present a chip that is the first mixed-signal reconfigurable array to do away with hard wired data converters, integrating digital and analog devices in such a fine-grained manner that data converters can be synthesized out of the array fabric, if needed, in the size, type, and quantity desired by the mixed-signal application, albeit at reduced performance over hardcoded converters. And while the number of digital and analog devices is certainly fixed, they share a common global routing resource, allowing a more digital or analog biased design to consume more of the common routing resources, which may lead to larger utilization of the fabric for circuits of varying digital to analog bias.

We present the design and verification of a mixed-signal, reconfigurable, floating-gate based array comprising a fine-grained interleaving of interchangeable field-programmable gate array (FPGA) based and field-programmable analog array (FPAA) based tiles, called the FPAADD: Field Programmable Array of Analog and Digital Devices. The general architecture is shown in Figure 1 and die photo in Figure 16.

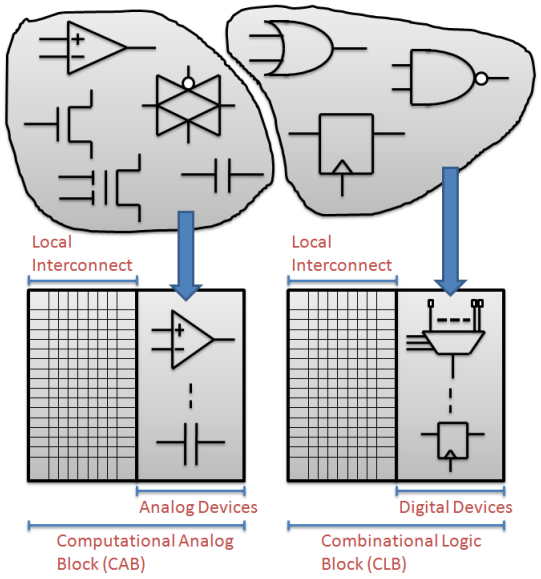
Low level analog devices (Figure 1a): floating-gate input operational transconductance amplifiers (FG-OTAs), MOSFETs, capacitors, transmission gates, and multiple-input translinear elements (MITEs) are grouped together with a sea of reconfigurable routing, local interconnect, for wiring the devices together within a tile, to make the computational analog blocks (CABs). While digital elements: arbitrary logic implementing look-up tables (LUTs), and flip-flops (FFs) are clustered together with local interconnect to make combinational logic blocks (CLBs).

A CAB or a CLB along with global interconnect, wiring that allows signals to propagate between and through tiles (Figure 1b), make up the analog and digital tiles, respectively. The tiles are designed to be interchangeable with respect to connectivity, with the majority of the global interconnect lines (tracks) being passive and able to propagate analog and digital signals, while a few tracks are reserved for buffered routing of analog or digital signals.

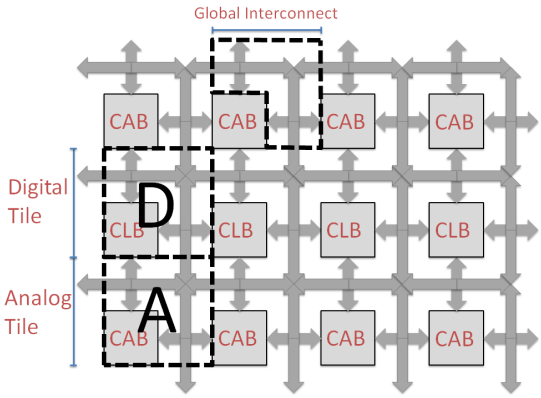
Floating-gates are used for the switches and state storing elements on the chip. The dynamic range of the FG switches allow for ON performance comparable to transmission gates with parasitic capacitance of a single FET, with leakage currents an order of magnitude less than standard SRAM based alternatives. The non-volatile nature of the floating-gates means the chip does not have to be reprogrammed on power up. The continuum between the ON and OFF states allow the routing infrastructure to perform useful functions other than just connecting nets: tunable delays, current biases, and vector matrix multipliers (VMM), for instance, are all easily implementable by the interconnect.

The core of the FPAADD is an array of these tiles. The tiles are interleaved on a row by row basis with a higher density of digital rows on the bottom and analog rows on the top. The rest of the chip is floating-gate selection and programming infrastructure (controlled by an SPI bus), and buffered and non-buffered I/O. The top level arrangement of the chip is shown in Figure 1c.

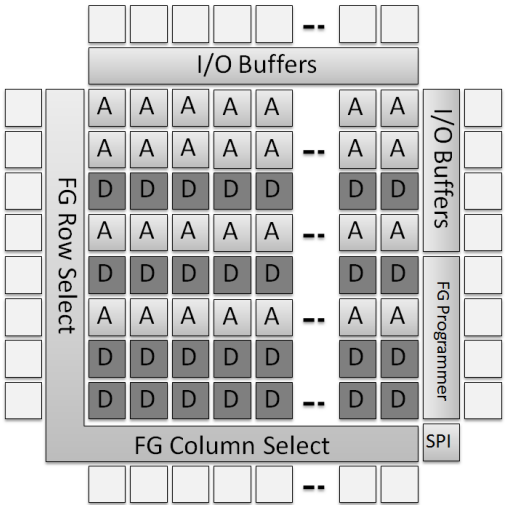
The rest of the paper is organized as follows: Section



(a)

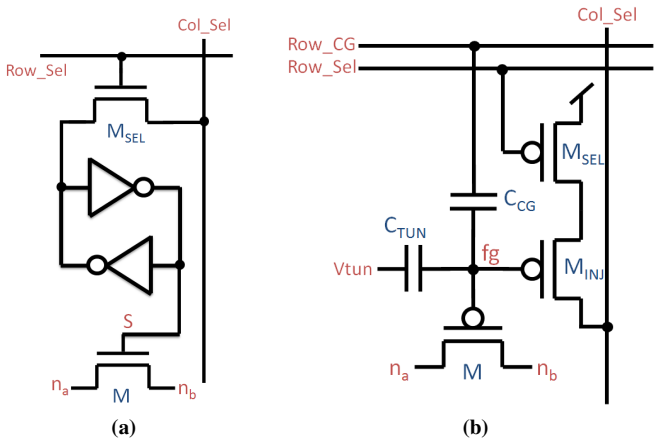


(b)



(c)

Figure 1 – The general architecture of the FPAADD: a) Left, analog devices (MOSFETs, capacitors, etc.) are grouped together with local interconnect, a sea of reconfigurable switches for connecting the devices together, to form Computational Analog Blocks (CAB). Right, digital devices (Flip-Flops and look-up tables) are grouped together with local interconnect to make Combinational Logic Blocks (CLB). b) Interchangeable digital and analog tiles are built from either a CLB or a CAB with reconfigurable routing that allows signals to propagate between tiles (global interconnect). c) Top level. Blocks not to scale.



(a) (b)

Figure 2 – a) Switch element M is an nFET with SRAM based state storage as is typical of modern FPGAs. $M's$ state is set by turning on M_{SEL} and driving $ColSel$ high or low. b) The FPAADD uses floating-gates for the switch elements and memory. Switch element M is a pFET whose gate has no DC paths under run-time bias, and whose state is stored as charge trapped on the fg node. Negative charge is added through M_{INJ} by channel hot electron injection or removed through C_{TUN} by Fowler-Nordheim tunneling

II describes the detailed architecture and the building of the FPAADD, Section III accounts the software stack used to place and route and program netlisted circuits onto the FPAADD, measurements of low-level computational elements, measurements of parasitic delay from switches, and general system verification is shown in Section IV, in Section V a a subset of potential applications and their mapping to the FPAADD are described as well as measured system results for two applications: a VCO based ADC, and a 2^{nd} order sigma-delta modulator are presented, and lastly, Section VI concludes the paper.

II. FPAADD Architecture

The introduction provided an architectural overview of the chip, here is presented an in depth description of the various components of the FPAADD.

A. Floating-Gate Switch

The most basic and ubiquitous component of any highly reconfigurable architecture is the switch and the switch's state storage. In the majority of modern FPGAs, this is implemented by a single nFET whose gate is driven by SRAM (Figure 2a). Figure 2b shows the corresponding element as used in the FPAADD; the switch and memory are implemented by a floating-gate pFET. In both figures, the transistor, M, is the switch element used in the circuit.

While the SRAM solution is very dense, the floating-gate based solution offers many advantages: on resistance comparable to that of a transmission gate while maintaining a single device's parasitic capacitance, lower leakage current for off devices, non-volatility, rail-to-rail operation, and for applications that desire it, the ability to precisely implement a resistance somewhere between the "on" and "off" states [9].

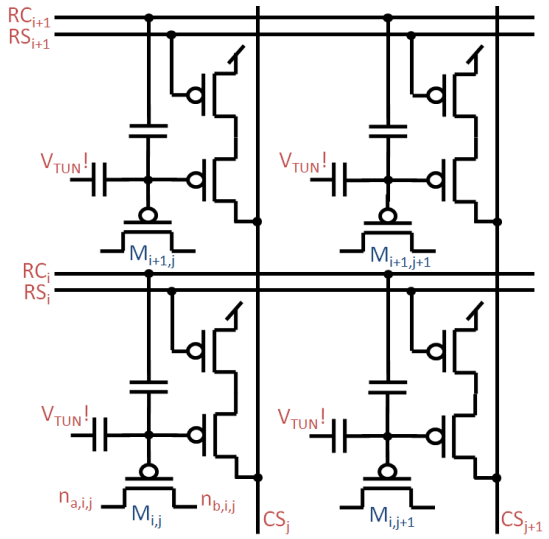


Figure 3 – Programming is achieved by globally removing charge from the floating-gate nodes through C_{TUN} via Fowler-Nordheim tunneling, and then selectively adding charge through M_{INJ} with impact carrier hot channel electron injection. Injection of charge per row is controlled by the selection lines CS_i, and per column by the drain lines CS_j. Since injection requires high channel current and steep electric fields, injection current is lowest in deep subthreshold and high above threshold and peeks somewhere in between. The RC_i lines are used to move the floating-gate coupled bias during program to push M_{INJ} back into a region more conducive to injection.

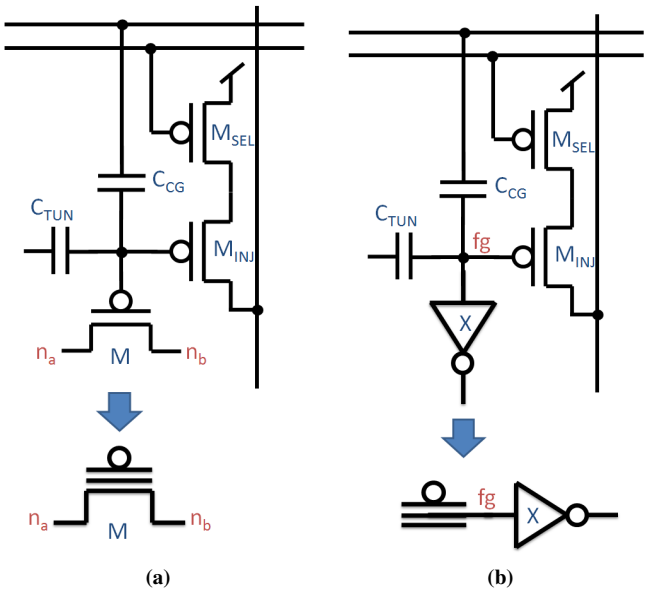


Figure 4 – a) pFET switch with floating-gate memory and circuit symbol. b) Circuit symbol for a floating-gate memory element setting the gate input voltage of an inverter.

Floating-gate transistors are normal MOSFET devices where the gate is completely insulated by silicon-dixode. This means that all terminals capacitively couple onto the gate, and the device's effective threshold is modified by charge trapped on the gate. The modeling of the behavior of this device can be achieved by substituting for the gate voltage the following in any MOSFET current equation:

$$V_g = \frac{1}{c_{tot}}(Q + \sum v_i c_i) \tag{1}$$

Where c_{tot} is the total capacitance at the floating-gate, c_i and v_i are the capacitance coupling into the floating-gate and the voltage at the i^{th} node of the device. Q is the net total charge on the floating gate that is modified (or programmed) to set the state of the device.

Since the gate has no dc path to ground, the charge Q is trapped and remains on the floating-gate during normal circuit operation. However, for the floating-gate to implement the state storage of the switches, this charge has to be modifiable in a selective manner.

Figure 3 shows an array of floating-gates. All floating-gates are erased by applying a very large voltage to the global signal $V_{TUN}!$ which enables electrons to flow off of the floating-gate nodes into the $V_{TUN}!$ net by Fowler-Nordheim tunneling. This makes all floating-gates very positive in potential, and effectively turns off all floating-gate pFETs. To selectively turn on a pFET floating-gate, electrons are injected onto the floating-gate by way of impact based channel hot electron injection through M_{inj} (Figures 2b and 3) [10]. If A_{VDD} is set to a high enough voltage to allow for injection, the i^{th} floating-gate is injected by setting RS_i low and CS_j low. Injection takes place in regions where the electric field is strong enough to heat a significant portion of the minority carriers in the pFET channel to energies high enough to conduct in the silicon-dioxide, and the field in the oxide is such that those carriers are attracted to the floating-gate. The highest electric fields are achieved by operating the device in subthreshold, where almost all of the source to drain voltage is dropped in a very short region near the drain, and setting the source to drain voltage as high as possible. In this case, the probability of injection of each carrier is maximized, but when this is maximized the amount of available carriers tends to be very low. Because of this, injection rate is maximized somewhere near the onset of above threshold, and tends to be very poor in regions of high above threshold (lots of carriers, not very high fields) and deep subthreshold (very high fields, but not many carriers) [10]. In order to bias the device in regions conducive to injection, and further optimize the dynamic range of programmability, there is a control gate on each floating-gate that is controllable during injection.

B. Computational Blocks

Building up the local interconnect and high level portions of the chip is greatly facilitated by defining some circuit symbols: Figure 4a shows the symbol used for a floating-gate pFET switch, and Figure 4b the symbol for when a floating-gate is used as the gate input to a larger circuit like an inverter.

An open circle, as shown in Figure 5a, denotes when a switch is used to connect two abutting net lines. When a switch is used to allow connectivity between two crossing net lines an open circle is drawn over the crossing of the two nets (5b). Figure 5c shows the symbol for an s-switch connection topology, an open square. The s-switch allows a signal entering

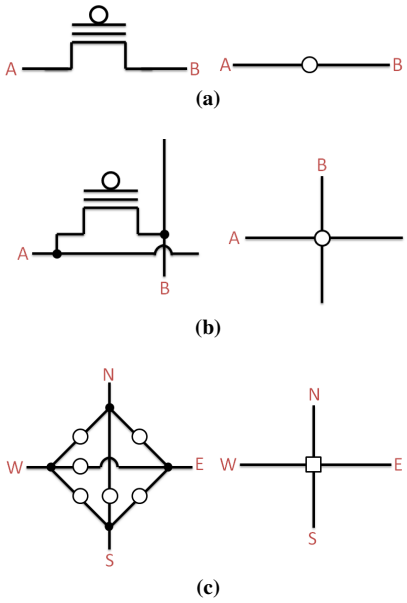


Figure 5 – The left column contains circuits and the right column their abbreviated symbols used in this paper: a) a pFET floating-gate switch connecting two abutting nets, b) a pFET floating-gate switch connecting two crossing nets, c) six pFET floating-gate switches implementing an s-switch.

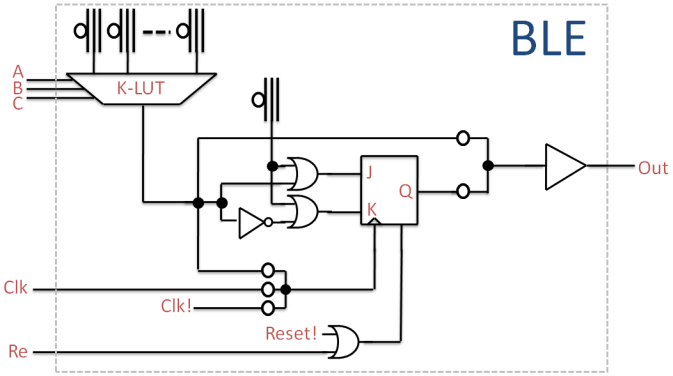


Figure 6 – The BLE is primarily a 3-input LUT whose output either gets registered or not with a FF. Our registering device is implemented as a JK-FF that is either configured as a standard FF or a T-FF, whose clock can come from the local interconnect, the output of the LUT, or a global line.

from any side to propagate across, make a turn, split in two directions, allows two nets to cross each other, or turn away from each other.

1) Combinational Logic Block: The Basic Logic Element (BLE) is the building block of the digital circuits. The standard BLE is a k-input look-up table whose output is either registered or not by a flip-flop. Shown in Figure 6 is the BLE implementation used in the FPAADD. Instead of using a standard flip-flop, a JK-FF is used that can be configured as a T-FF or a D-FF. The clock can be routed from the local interconnect, the BLE's look-up-table, or come from a global signal. These choices were made to allow of high density synthesis of asynchronous counters.

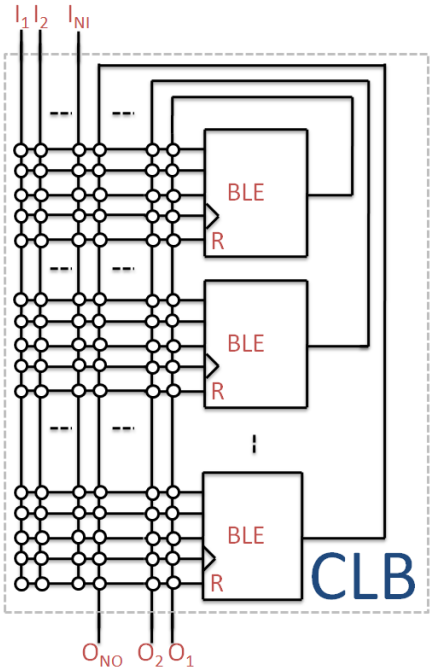


Figure 7 – The CLB comprises BLE devices and a sea of local interconnect. The outputs from the NO number of BLEs are the primary outputs from the CLB, and the inputs to the BLEs come from the NI number of primary CLB inputs and the NO BLE outputs. In the FPAADD $NO = 4$ and $NI = 8$.

Figure 7 shows that the CLB is comprised of NO number of BLEs and a sea of local interconnect. The inputs to each BLE come from either any of the NI primary inputs to the CLB or from the outputs of any BLE in the CLB. The NO outputs of the CLB are hardwired to the outputs of the BLEs in a one-to-one fashion. The configuration of the local interconnect allows for a deterministic and guaranteed routing solution for any clustering of any NI inputs and NO BLEs. Where $NO = 4$ and $NI = 8$.

2) Computational Analog Block: The CAB is the analog equivalent of the CLB. It is a cluster of analog devices and local interconnect, however, instead of a homogeneous set of devices, the CAB in the FPAADD contains: floating-gate based operational transconductant amplifiers (OTAs), switched capacitor optimized transmission gates, MOSFETs (either common centroid pFETs or nFETs), capacitors, and multiple-input translinear elements (MITEs: floating-gate pFETs with multiple input control gates). This set of devices was chosen to make the FPAADD CABs compatible with the generic CABs of the RASP2.9a chip [4]. Inputs to the devices come from the NI primary inputs, the two hardwired V_{DD} and gnd signals, or the outputs of any device in the CAB. The NO outputs of the CAB are multiplexed from the set of CAB device outputs. This was chosen because the number of devices in the CAB exceeded that in the BLE and it was desired to keep the same number of I/O in the CAB as in the CLB: $NO = 4$ and $NI = 8$.

While the routability of the CLB was complete, this is not quite the case for the CAB. Whether or not there exists a completely deterministic and guaranteed routing solution for

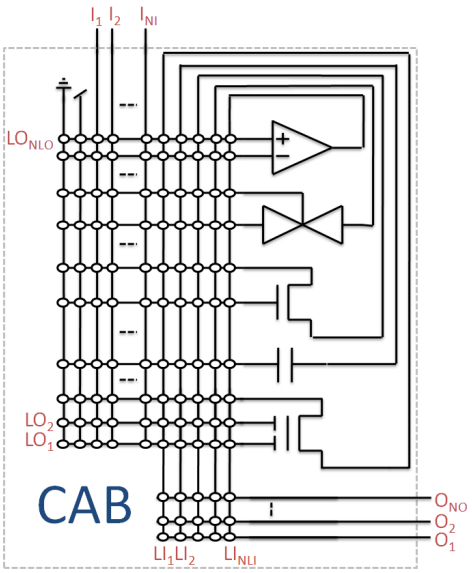


Figure 8 – CAB architecture showing devices and local interconnect. Net name terminology derived from the perspective of the local interconnect. Inputs to the local interconnect are vertical lines and outputs from the local interconnect are horizontal. I and LI are the primary inputs to the CAB and the outputs from the CAB devices respectively. O and LO are the primary outputs from the CAB and inputs to the CAB devices respectively. There are exactly twice as many devices in the CAB as shown, and $NO = 4$ and $NI = 8$.

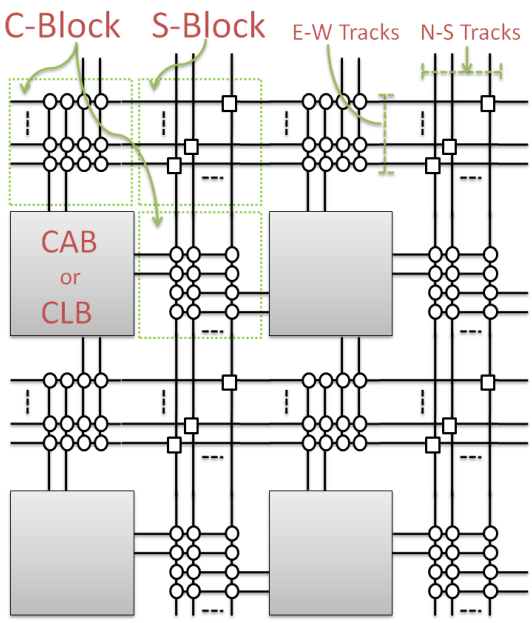


Figure 9 – The global interconnect comprises vertical and horizontal track segments isolated by S-Blocks. The S-Blocks allow signals on tracks to propagate to neighbor tracks or to change directions. The C-Blocks provide connectivity from the global tracks to the primary inputs and outputs of the CLBs and CABs.

all combinations of NI inputs, NO outputs and CAB devices depends on whether or not the clustering can be partitioned such that the implied input/output relationship of the devices is preserved: an output can go to multiple inputs, but multiple outputs can not go to a single input. In the CLB, the inputs and outputs are well defined, as is the case with CMOS digital gates. While an OTA may have well defined inputs and outputs, and the gate of a MOSFET is easily classified as an input, classifying the sources or drains of a MOSFET, for instance, as either inputs or outputs is rather arbitrary. If a partitioning of the circuit to be clustered in the CAB preserves these mappings then the cluster is guaranteed to route in a deterministic manner. Since many analog circuits do not partition this way, this does not automatically mean they will not route, the output multiplexor allows for limited support of shorting of outputs. If outputs are to be shorted, and if the output is also a primary output, then the output multiplexor can handle this. The only time output shorting will fail is if two devices are to short their outputs, and this net does not propagate out of the CAB or to the input of any device in the CAB, and all CAB output lines are already occupied with other nets.

C. Global Interconnect

The global interconnect follows a standard track based scheme with connection blocks (C-Blocks) getting inputs and outputs out of the CABs and CLBs and onto the tracks. Switch blocks (S-Blocks) allow track segments to be connected across, or to make turns. Figure 9 shows a two by two array of tiles where each tile contains either a CAB or CLB and global interconnect: two C-Blocks and an S-Block. There are 11 tracks in the north-south direction, and 11 in the east-west direction.

The C-Blocks in the FPAADD are implemented as a completely populated floating-gate matrix. The C-Blocks are not fractional, all inputs and outputs from the computational blocks have access to every track, and all track segments span one tile length.

The S-Blocks are a diagonal arrangement of s-switches (one buffered, ten passive) that allow signals to propagate across or to change directions into neighboring tiles, but the diagonal nature keeps the signals on the same track number as they started. The standard s-switch is implemented as shown in Figure 5c, which passively passes both analog and digital signals. Every s-switch is of this passive type except for the bottom left ones on the first track.

These s-switches on the bottom track are buffered. Each digital tile's S-Block has a single digital buffered s-switch and each analog's a single analog buffered s-switch. Two different buffered s-switch topologies can be seen in Figures 10 and 11 . Both circuits are bi-directional, and allow for the same direction choices of signal propagations as the passive s-switch. The first circuit uses significantly less switches, has less internal parasitics per track, forces all entering signals to leave buffered, and requires four buffers. The second circuit is basically a passive s-switch with the ability to insert a single buffer on the input from one of the directions. It requires more switches than the first, has more internal parasitics, can buffer either one or zero signals, has the same input capacitance as a passive s-switch, and uses only one buffer. In general, the first topology will be faster, but larger than the second

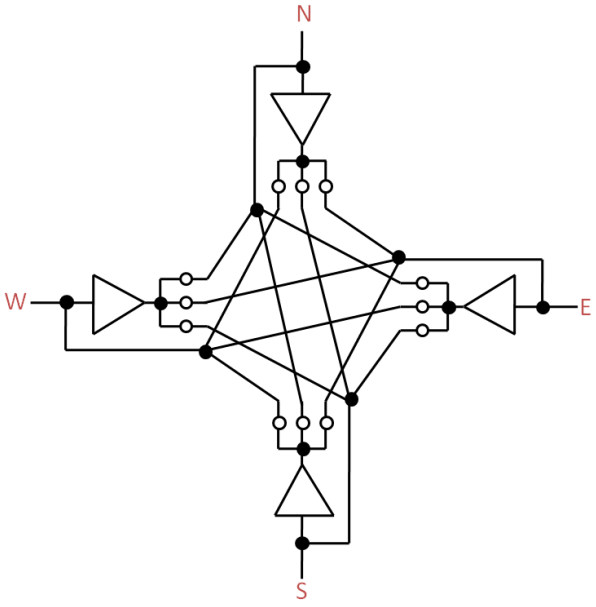


Figure 10 – The s-switch topology used in the digitally buffered s-switches. This circuit is bi-directional and allows for the same routing options as the passive s-switch. The four digital buffers are simple inverter chains.

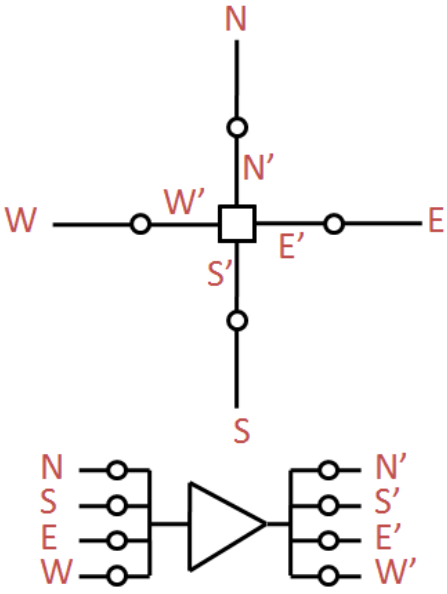


Figure 11 – The analog buffered s-switch topology. This circuit wraps a standard passive s-switch with hooks allowing the signal to pass through an analog buffer (a simple unity gain 9-T OTA buffer). This topology requires only one buffer, allows a signal to pass unbuffered, and while it can pass multiple signals, it can only buffer one of them.

Array size	27x8: 108 digital tiles and 108 analog tiles.
Chip IO	33 generic IO pads, 11 digitally buffered bi-directional pads
Devices per CAB	2 FG-OTAs, 2 TGATEs, 2 Capacitors, 2 FETs (nFET or pFET), 2 MITEs
Devices per CLB	4 BLEs
CAB / CLB I/O	8 inputs, 4 outputs
BLE	3-input LUT, routable clock and reset, reconfigurable for asynchronous adders
C-BLOCK	11 Total tracks, all of segment length 1, fully connected connection blocks
S-BLOCK	diagonal with 1 digital or analog buffered s-switch per tile on the first track
Process	CMOS 0.35um Double-Poly, 4-M
Voltage	2.4V at runtime

Table I – FPAADD specifications

topology. Because the analog buffers (simple 9T floating-gate programmable OTA based unity-gain buffers) are much bigger than the digital ones (two-stage inverter chain) we chose the second topology for the analog buffered s-switches and the first topology for the digital buffered s-switches.

Table I contains a list of specific parameter values used in the FPAADD.

D. Architectural Comparison

The CAB devices, floating-gate design, and floating-gate programming infrastructure were all derived from the RASP 2.9a chip, a next generation FPAA from the line developed by Hasler *et. al.* [11], [12], [4]. Significant differences from the RASP 2.9a include the choice of a Manhattan style global routing architecture, and a feedback output local interconnect scheme. The global interconnect has significantly less parasitics over short distances than the RASP's global scheme, where global tracks span the entire length of the chip. There are buffers in the global interconnect whereas the generic RASP line contains none. The local interconnect of the FPAADD has 30% lower parasitic capacitance and 50% less parasitic resistance between routed CAB devices in the local interconnect (devices in the FPAADD can be connected with one switch, but in the RASP chips require two at minimum) at the cost of decreased routeability described earlier. In the RASP line, CAB devices were disconnected from the routing infrastructure during program time with large transmission gates in order to not expose the devices to injection level programming voltages, but with careful circuit consideration, these can be removed in almost all cases.

The global interconnect scheme as well as the local interconnect and CLB devices draw heavily from standard Manhattan style FPGAs [13], [14]. Semi-arbitrary design decisions regarding architectural parameters, such as number of tracks, cluster size, placement of buffers, etc. aside, the digital tiles look very similar to previously made FPGAs. The biggest difference being replacing the switch elements and SRAM with programmable floating-gate pFET transistors[11], [12], [4].

Floating-gates are very similar in operation to non-volatile technologies such as EPROM, EEPROM, and FLASH; various

FPGAs and CPLDs have been built using these technologies [15],[16],[17]. The floating-gates transistors in the FPAADD are built in a standard CMOS process. They have a higher dynamic range of programmed voltage leading to significant performance increases in power, speed, and signal integrity at the cost of density compared to conventional EEPROM and FLASH devices.

In [15], they claim that switching from using the EPROMs as the actual switch to simply using the EPROMs to control the gate of CMOS devices, that a 10x speedup was achieved. This is similar to the problem that pass-transistor logic, often used in FPGAs, face when trying to pass a logic-level (VDD for nFETs and GND for pFETs) that causes the devices to enter subthreshold before completely passing the signal. This results in signal levels about a threshold voltage in the wrong direction after reasonable amounts of time, significant speed degradation, and can lead to an exponential increase in leakage current in gates driven by these logic levels. The range of floating-gate programmability is higher than a threshold voltage above VDD and a threshold voltage below VDD, because of this the devices stay in above threshold while passing the whole rail-to-rail voltage. Small signal resistance sweeps in [12] show floating-gate switches being as good as transmission gates but at half or better parasitic capacitance.

III. CAD Software

Much work has been done in the realm of synthesis for FPGAs; many software packages are available from industry and the open-source community alike. The field of synthesis for FPAAs, however, is far from mature. While the algorithms for placement and routing certainly have application to FPAAs, what does not translate so well are the cost functions (other than trivial ones like area and routeability) to evaluate the desire-ability of routable solutions. FPGA synthesis is largely timing driven, where propagation delay models are used to identify the worst case delay of the critical path (further effort can be spent to then reduce the amount of devices on non critical paths for power optimization). While line delays certainly have some application to analog circuits, they are by no means the appropriate metric for all circuit nets.

In [18], [19] the authors successfully apply standard placement and routing algorithms to map analog circuits to FPAAs with global parasitic reduction being the metric of choice, later work is done to extract parasitic models to back annotate the initial input spice netlist for simulation, with fitness evaluation and iteration up to the user. The strategy in [20], [21], [22] is to partition the mixed signal reconfigurable system into the digital and analog subcircuits at data converter interfaces and apply different cost functions to each. Models for SNR estimation are developed that start with known device SNR and its degradation by connection topology of interconnect: cascode, fan-out, fan-in, and feedback. Bandwidth is also estimated using the data converter's Nyquist criterion as the bottleneck.

None of these approaches take into account the appropriateness of applying different cost functions to different net types. For instance, an algorithm that places negative weight on average parasitic capacitance of all nets will inefficiently try to reduce parasitic capacitance on nets that are insensitive to it, like internal nets of DC bias generators.

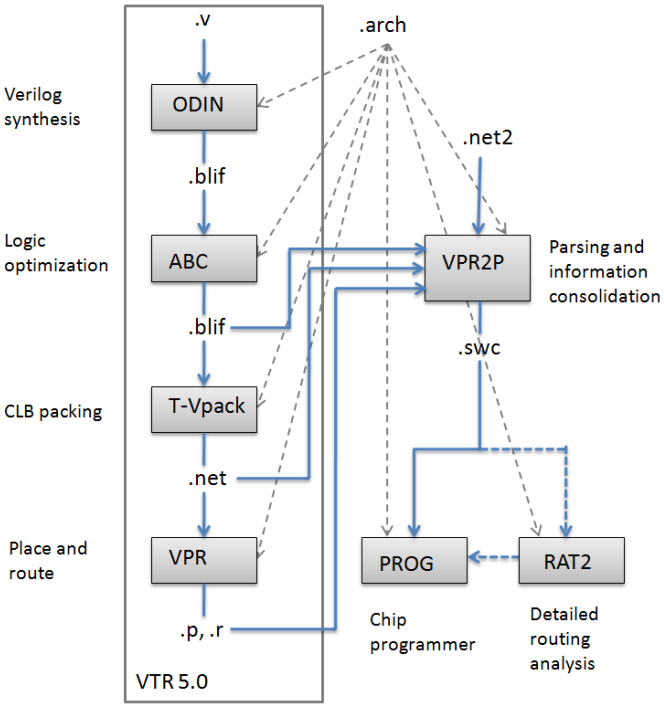


Figure 12 – The software stack used for programming the FPAADD. From the VTR flow: ODIN takes an input verilog file and performs logic synthesis targeting LUTs, FFs, and macro function blocks. ABC performs logic optimization. T-Vpack clusters LUTs and FFs into CLBs. And VPR places and routes the result. VPR2P takes an input describing the internal configuration of the CABs that are treated as black boxes in the VTR flow, and all of the intermediate outputs of the VTR flow, and creates a switch list. The switch list can be directly programmed or analyzed and modified by the detailed routing analysis tool, RAT2. All programs in the flow take various pieces of architectural descriptions of the target system.

The software suite, VTR, was extended to perform placement and routing on the FPAADD. As of writing, the flow is completely area driven.

A. Verilog To Routing

Verilog To Programming (VTR) is an open source, academic software suite that given an input verilog circuit description and an input FPGA architectural description, performs synthesis and place and route (Figure 12). The suite consists of the following programs: ODIN II, which performs logic synthesis to standard cells (in this case, LUTs, FFs, and macro functions) [23], ABC which performs logic optimization [24], and T-Vpack and VPR which perform packing of LUTs and FFs into CLBs and then placement and routing [25].

While the flow supports synthesis of verilog to the standard FPGA building blocks, it also supports the targeting of larger functions that may exist as dedicated hardware blocks on a heterogeneous FPGA. For instance, it is often common to include hardware adders or multipliers in FPGAs as the

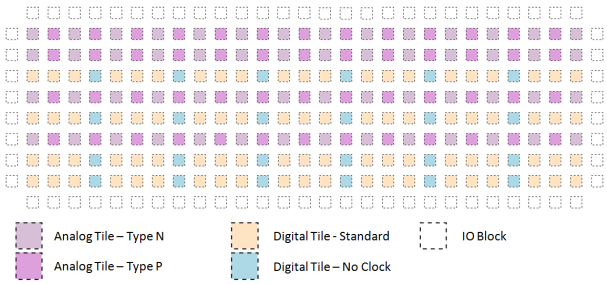


Figure 13 – The FPAADD architecture as understood by VPR. There are digital rows and analog rows. Every other analog tile in an analog row has either a set of matched pFETs or matched nFETs. Every fourth digital tile in a row has no routable clock and is designed for synthesis of dense asynchronous adders. Only the standard digital tiles are treated as non-black boxes.

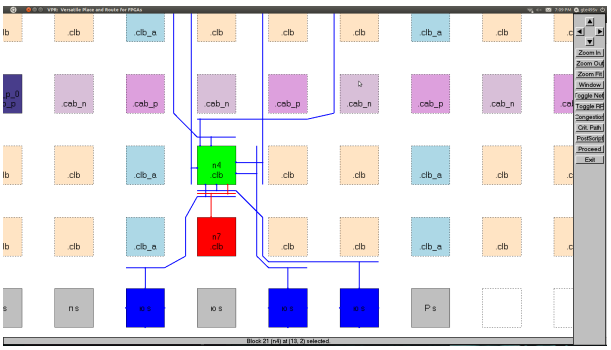


Figure 14 – Screen shot from the VPR GUI displaying global placement and routing information. Highlighted are the input and output nets from the selected green tile.

synthesis of these rather common functions are often the bottlenecks in an FPGA implemented circuit design.

The support of black boxes made VTR a very attractive starting point in creating a software chain to provide placement and routing on the FPAADD. Digital circuitry could be synthesized all the way from verilog while the analog circuitry could be treated as black boxes and simply placed and routed.

B. Routing on the FPAADD

The architecture of the FPAADD, as understood by VPR, is shown in Figure 13. While VPR will route circuits to arbitrary architecture graphs, it also supports a robust and scalable XML based architecture description language for quick graph building. Since the FPAADD was designed with a Manhattan style global and local interconnect scheme, describing the FPAADD in the VPR architecture language was relatively straight forward. Only a few minor modifications to VPR 5.0 were necessary.

The current flow starts with the circuit input as a blif and a net2 file. The .blif file contains all of the digital circuitry as described as netlists of LUTs and latches with black boxes for the analog circuits. T-Vpack then packs the digital circuits into CLBs and the CABs are already prepacked in the net2 file. VPR then places and routes the CLBs and CABs.

The program VPR2P was written to take all of the intermediate file outputs of the VTR flow, consolidate the information, fill in the blackboxes with information from the .net2 file,

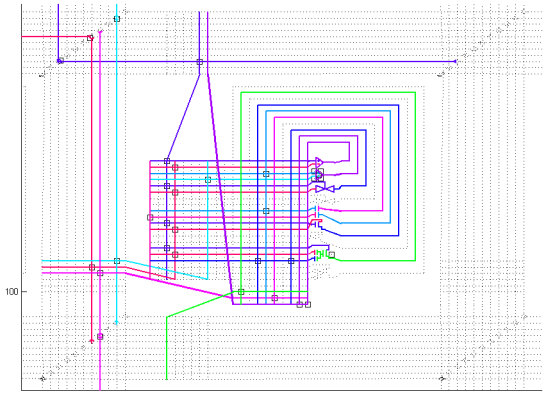


Figure 15 – The RAT2 tool displays detailed low level routing information. Switches can be set by hand with the tool.

and then to translate the information into the corresponding physical switch locations on the FPAADD. The output is a row column switch list that is input into our programming software that handles chip and board communication as well as the algorithms for erasing and programming the floating-gate memory elements.

Figure 14 shows the VPR GUI for some small benchmark targeting the FPAADD. The figure shows a highlighted CLB, in green, with its routed input and output nets, blue and red, respectively. Since VPR is not concerned with local interconnect, as routing at that level is deterministic, the GUI does not show the internals of the blocks. So in order to analyze the detailed routing solutions, global routing and local routing, and as well as to provide for a way to set and unset switches by hand, the RAT2 tool was created.

A screenshot of RAT2 showing a portion of a routed circuit on a portion of the FPAADD is shown in Figure 15. The RAT2 is a simple program written in MATLAB that can read in a switch list, display the routing solution, modify by means of a point and click interface the switch list, and dump out a switch list. This switch list can then be used to program the chip.

IV. Low-Level Component & System Verification

The FPAADD as described in Section II was fabricated in a standard double-poly, 0.35um process. A die photo of the FPAADD is shown in Figure 16. The system is operated at 2.4V during run time, as opposed to 3.3V, to increase retention of the stored charge on all floating-gate transistors [4].

All CAB and CLB devices, as presented in Table I, are verified to be functional, via successful interconnect routing to I/O pads; global interconnect, local interconnect, and interconnect buffers are all working as expected. Simple circuits have been built: XOR gates and full-adders implemented in the CLB floating-gate based LUTs, asynchronous adders generated from FFs and LUTs, MOSFET threshold and characterization data extracted from routed out fet devices in the CABs, and ring oscillators built out of the buffered global interconnect. Verification and programming of the the floating-gate transistors was performed and performance was

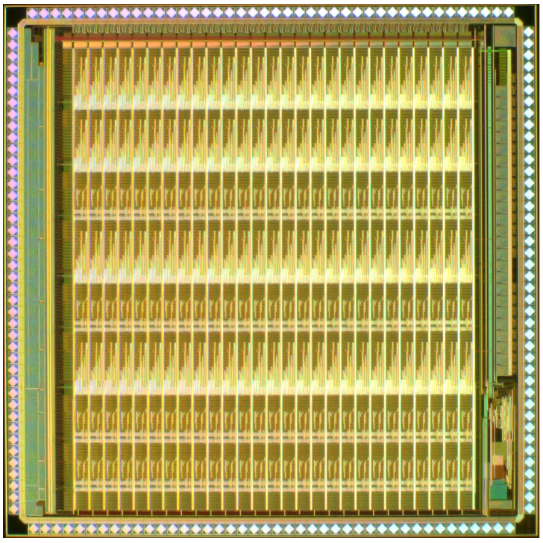


Figure 16 – Die photo of the fabricated FPAADD.

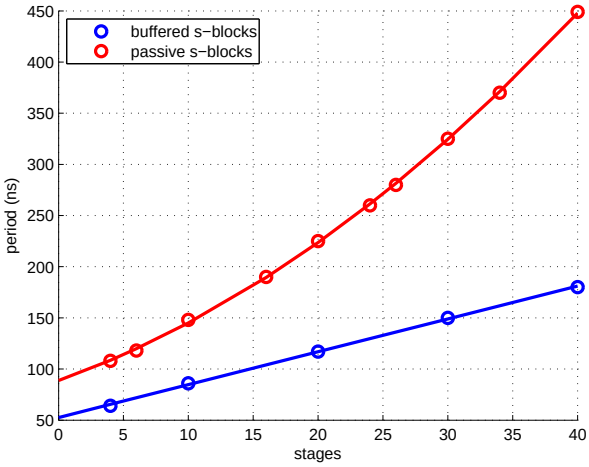


Figure 17 – Ring oscillator period versus number of additional interconnect stages (s-block to s-block) for digitally buffered and passive s-blocks. The incremental delay due to a digitally buffered s-block is 1.6ns.

the same as in [4]. The components for the CAB were taken from previously designed FPAAs. Performance metrics for the individual components found in the CAB can be found in Basu, et, al 2000, [4].

To evaluate the performance of the interconnect, digital ring oscillators were built of varying depth. Each additional stage comprises of a C-BLOCK to S-BLOCK connection, where each oscillator uses entirely digitally buffered s-switches or passive s-switches. In Figure 17, oscillator period is plotted as function of the number of additional stages. As expected, the delay of the oscillators using non-buffered s-switches increases quadratically with the number of stages as is typical of RC ladders, and the buffered ones increase linearly. The delay of moving from one tile to the next through a digitally buffered s-switch is 1.6ns. Using a similar method, the BLE to BLE delay was measured to be less than 7ns.

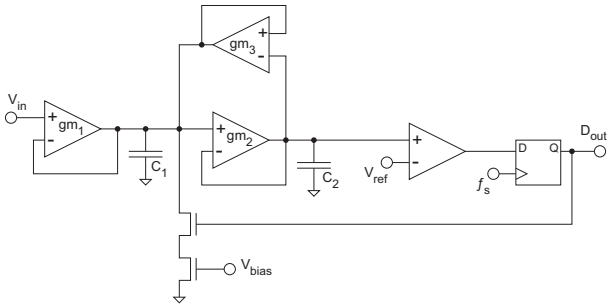


Figure 19 – A 2^{nd} order sigma-delta modulator with 1-bit DAC feedback.

V. Example FPAADD Application & Results

Previous FPAAs have been used to build continuous time filters, vector matrix multipliers, AM receivers, analog speed processors, among others [3], [4]. The reconfigurable and mixed-signal nature of the FPAADD allows the user to address a variety of applications from pure analog to mixed-mode to pure digital with circuit complexity ranging from small to large, including all FPAA applications in reported literature. Two example systems applications have been built to demonstrate FPAADD application performance: a VCO-based ADC and 2^{nd} order low-pass sigma delta modulator.

A. VCO-ADC

Voltage controlled and current controlled oscillators were built using CAB components. Asynchronous counters, state machines, decimators, and registers were built out of the CLBs. Combinations of these components were arranged to build an 8bit VCO based ADC (Fig. 18), and a single loop low-pass, second-order delta sigma modulator with a one-bit quantizer was also built using a mix of digital and analog components (Fig. 19).

Figure 20 shows the frequency versus control voltage plot of the VCO from Figure 18c. The dynamic range of the VCO was from 0.18mHz to 7Mhz, and when operated as a current controlled oscillator was linear up to 1Mhz. Higher than 1Mhz, while still operational, the OTA's delay becomes non-negligible and the input currents discharge the capacitance before the output can swing full rail, leading to an increase in frequency.

A VCO based ADC with 8-bit digital out was created to verify system performance. The digital back-end was clocked externally at 2Mhz (though the backend was operational up to 18Mhz), and a 200.137Hz, $0.4V_{PP}$ input sine wave was applied. The ADC was measured to have no missing codes, and its operation can be seen in Figure 21. INL and DNL data is not presented due to the non-linearity inherent in VCO based ADCs. The non-linearity of the ADC is due to the following effects: the input voltage to current converter is a simple nFET operated in subthreshold, so an exponential voltage to current conversion is expected, while the current controlled oscillator performs a linear conversion of input current to frequency, and the digital backend, instead of frequency, the period of the VCO is measured by means of counting a global clock between input pulses. Using the expected circuit behavior,

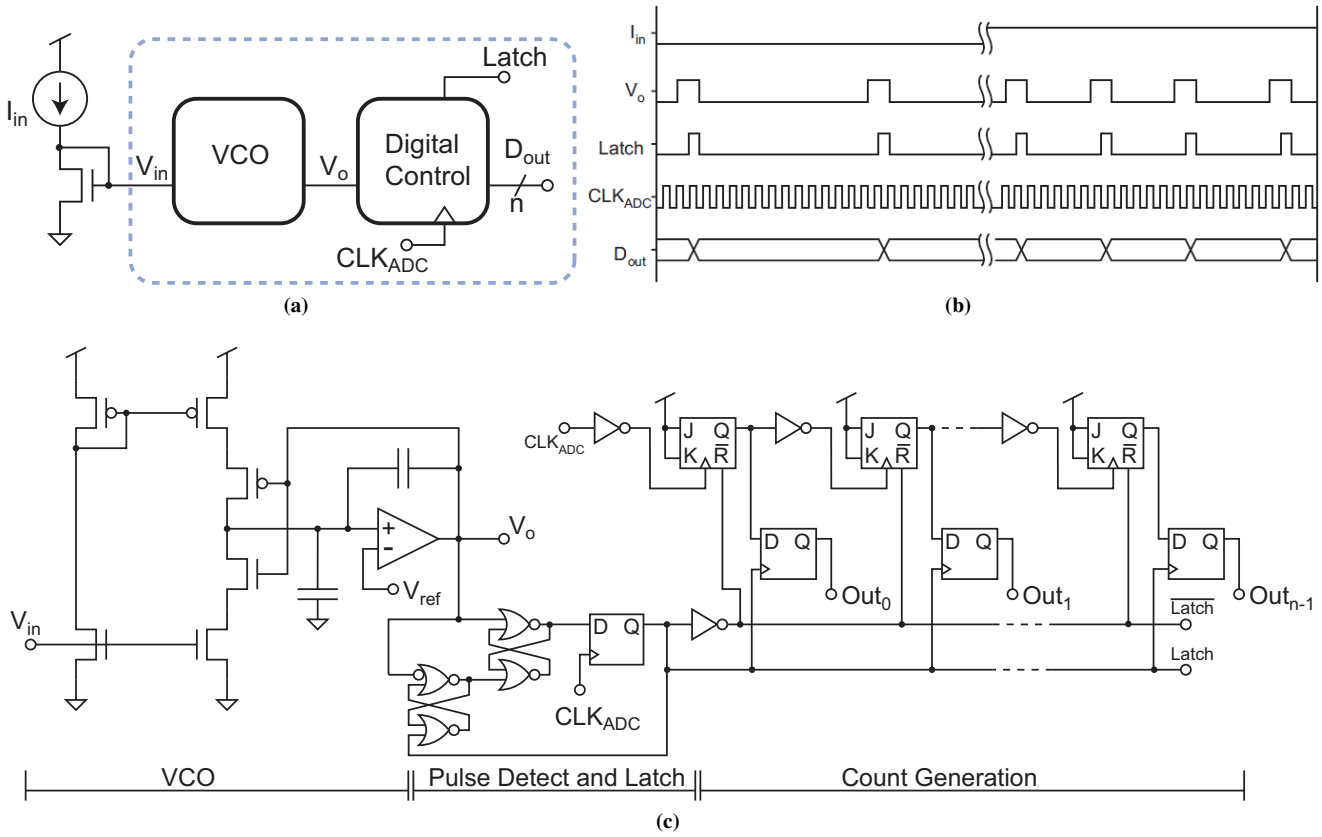


Figure 18 – An 8-bit ADC built on the FPAADD. a) Block Diagram: A current or Voltage Controlled Oscillator's (VCO) output period is measured by a digital backend. b) Timing diagram for the circuit's operation. c) VCO, pulse detection circuit and state machine, asynchronous counter and latches.

the input is reconstructed from the output by fitting it to the following equation:

$$-ln[aT_{out} + b] = V_{in} \tag{2}$$

Where T_{out} is the measured output a and b are terms lumping subthreshold parameters of the input I-to-V input stage and the linear current controlled oscillator stage. The red line is then reconstructed from the output and shows the circuit to be in excellent agreement with expected circuit behavior.

The VCO based ADC system consumed a total of 10 tiles (four analog and six digital) representing 4.6% of the total number of tiles in the FPAADD array. The percentage of device utilization within the six digital tiles was 88% while the utilization within the four analog tiles was 23%. Low element utilization of the CAB is due to the heterogeneous nature of the devices present within the CAB. The VCO used primarily discrete transistors found in the CAB along with an OTA and 2 capacitors leading to the low utilization value.

B. Delta-Sigma Modulator ADC

Figure 19 depicts the system diagram of a 2^{nd} order low-pass sigma delta created in the FPAADD. The low-pass filter was built using components from a total of 2 CABs, and a single CLB is utilized for the D Flip-Flop. The poles of the loop filter are designed to be located at zero. The sigma-delta modulator has a measured SNR of 24.1 dB and SFDR of 39.2 dB at a bandwidth of 20 kHz and sampling frequency of 2.5 MHz. Figure 22 is a 32k FFT of recorded data taken from the FPAADD at the previously stated input and sampling frequencies. Insufficient gain of the loop filter is the probable reason for lower than expected SNR. Further optimization of the loop filter is required to increase the SNR.

VI. Conclusion

A mixed-signal heterogeneous tile array (FPAADD) of CAB and CLB components has been built and presented. Verification testing of the system was performed at the component, tile, and system level. Initial results of the FPAADD display 7ns BLE to BLE performance and 1.6ns buffered tile to tile delay. Oversampling ADCs were implemented to test the functionality of the tile array and show the reconfigurable nature of the chip. The goal of the FPAADD is a bridge towards embedded systems containing the reconfigurabilty of a FPAA and digital processors. Thus, the next stage of research will be done in the areas of automated system routing, building larger systems that take full advantage of the computation properties of the FPAADD, and integrated digital processors allowing complete embedded systems.

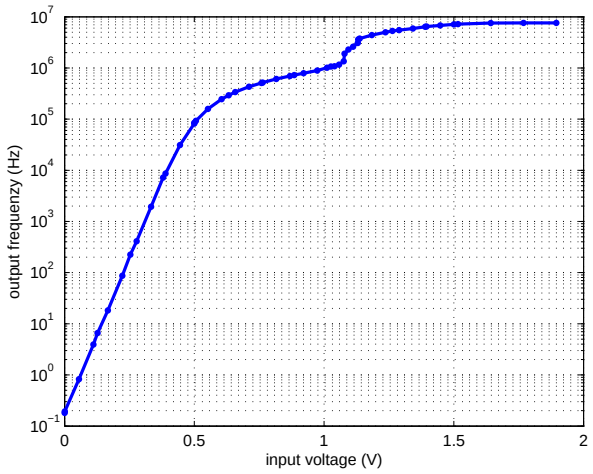


Figure 20 – Measured response of the VCO over varying input voltage.

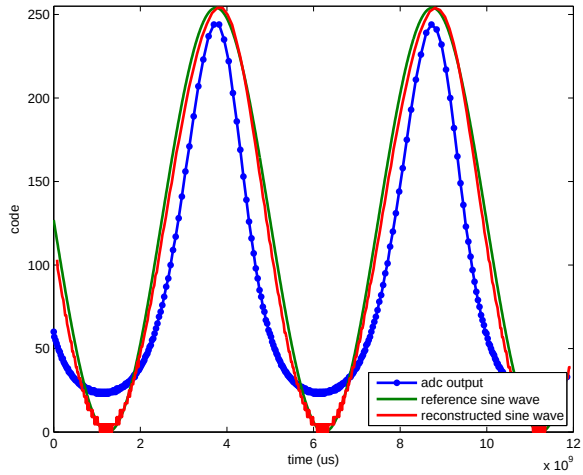


Figure 21 – 8-bit VCO based ADC digital output (dotted line) for a 200.137Hz input sine wave of $0.4V_{PP}$ and the reconstructed input signal.

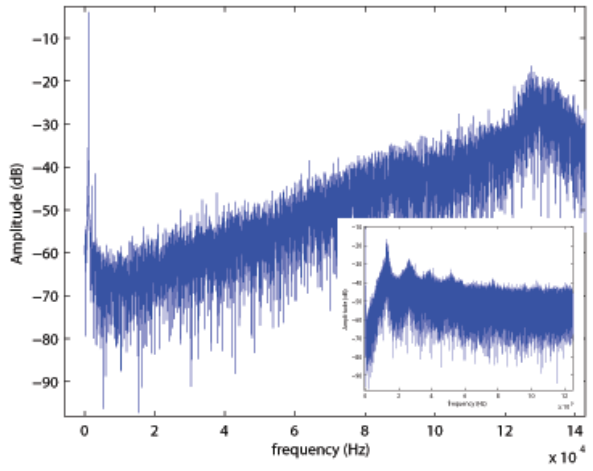


Figure 22 – A 2^{nd} order sigma-delta modulator with 1-bit DAC feedback. Measured power spectrum for an input of 1.0478 kHz at 2.5 MHz oversample frequency.

REFERENCES

[1] K. F. E. Lee and P. G. Gulak, "A transconductor-based field-programmable analog array," in *ISSCC Digest of Technical Papers*, Feb. 1995, pp. 198–199.

[2] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun, "A field programmable analog array for cmos continuous-time ota-c filter applications," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 2, pp. 125–136, 2002.

[3] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A field-programmable analog array of 55 digitally tunable otas in a hexagonal lattice," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 12, pp. 2759–2768, 2008.

[4] 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," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 9, pp. 1781–1794, 2010.

[5] *AN13x series AN23x Series AnadigmApex dpASP Family User Manual*.

[6] S. Ganesan, "Synthesis of mixed-signal systems based on rapid prototyping," 2001.

[7] J. Madrenas, J. M. Moreno, E. Canto, J. Cabestany, J. Faura, I. Lacadena, and J. M. Insenser, "Rapid prototyping of electronic systems using fipsoc," in *Proc. 7th IEEE Int. Conf. Emerging Technologies and Factory Automation ETFA '99*, vol. 1, 1999, pp. 287–296.

[8] P. Chow and P. G. Gulak, "A field-programmable mixed-analog-digital array," in *Proc. Third Int. ACM Symp. Field-Programmable Gate Arrays FPGA '95*, 1995, pp. 104–109.

[9] J. Gray, C. Twigg, D. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pfet switches in an fpaa crossbar network," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 468–471.

[10] P. Hasler, A. Basu, and S. Kozil, "Above threshold pfet injectionmodeling intended for programmingfloating-gate systems," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, pp. 1557–1560.

[11] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, and D. V. 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, 2005.

[12] C. M. Twigg and P. Hasler, "A large-scale reconfigurable analog signal processor (rasp) ic," in *Proc. IEEE Custom Integrated Circuits Conf. CICC '06*, 2006, pp. 5–8.

[13] A. M. Vaughn Betz, Jonathan Rose, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[14] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," vol. 81, no. 7, pp. 1013–1029, 1993.

[15] S. V. B. Ahanin, "A high-density, high speed, array-based erasable programmable logic device with programmable speed/power optimization," ACM International Workshop on FPGA, February 1992.

[16] P. Leventis, B. Vest, M. Hutton, and D. Lewis, "Max ii: A low-cost, high-performance lut-based cpld," in *Proc. Custom Integrated Circuits Conf the IEEE 2004*, 2004, pp. 443–446.

[17] C. Hu, "Interconnect devices for field programmable gate array," in *Proc. Int. Electron Devices Meeting Technical Digest*, 1992, pp. 591–594.

[18] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Placement for large-scale floating-gate field-programable analog arrays," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 8, pp. 906–910, 2006.

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

[20] S. Ganesan and R. Vemuri, "Analog-digital partitioning for field–programmable mixed signal systems," in *2001 Conference on Advanced Research in VLSI*, E. Brunvand and C. Myers, Eds. IEEE Computer Society, March 2001, pp. 172–185.

[21] ——, "Behavioral partitioning in the synthesis of mixed analog-digital systems," in *Proc. Design Automation Conf*, 2001, pp. 133–138.

[22] ——, "Technology mapping and retargeting for field–programmable analog arrays," in *DATE 2000 Proceedings: Design, Automation and Test in Europe Conference 2000*, Mar. 2000.

[23] P. A. Jamieson and K. B. Kent, "Odin ii: an open-source verilog hdl synthesis tool for fpga cad flows (abstract only)," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, ser. FPGA '10.

New York, NY, USA: ACM, 2010, pp. 288–288. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723176

[24] "Berkeley logic synthesis and verification group, abc: A system for sequential synthesis and verification, release 70731. http://www.eecs.berkeley.edu/ alanmi/abc/."

[25] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. Fang, and J. Rose, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2009, pp. 133–142. [Online]. Available: http://portal.acm.org/citation.cfm?id=1508150