Synthesis of Analog and Mixed-Signal Circuits on a Programmable Array

Ziyi Chen and Ioannis Savidis, *Senior Member, IEEE*

***Abstract*—In this article, a novel field-programmable analog array (FPAA) has been developed for the configurable implementation of various analog circuits. The proposed architecture not only supports system-level reconfiguration but also enables transistor-level programmability. The FPAA is comprised of a 3 × 4 configurable analog block (CAB) array, with a single configurable logic block (CLB) added to each column to allow for the programming of digital circuits. Passive devices, including programmable capacitors and resistors, and active transistor pairs (TPs), are utilized to implement both continuous-time and discrete-time circuits. A placement algorithm is developed that efficiently maps analog circuits onto the FPAA fabric by finding the optimal vertical and horizontal locations for the assignment of transistors. In addition, to reduce the complexity of placing devices on the fabric, a technique is developed that matches TPs in the same vertical level to predefined topologies in a library. Routers are included to connect devices implemented on the FPAA fabric. The proposed FPAA occupies an area of 4 mm^2 in a TSMC 65-nm fabrication process. The smaller circuits implemented on the FPAA fabric include a folded-cascode amplifier, a strongArm comparator, a continuous-time integrator, and a switch-capacitor integrator. The larger analog and mixed-signal circuits implemented on the FPAA fabric include a four-stage pipeline analog-to-digital converter (ADC) and a first-order delta-sigma modulator. The programmed folded-cascode amplifier exhibits a tunable gain of 28.3 dB to 34.8 dB and a programmable 3-dB bandwidth of 3.3 MHz to 5.3 MHz. The configured comparator provides a resolution of less than 3 mV when comparing two signals. The implemented first-order delta-sigma modulator operates at a frequency of 15 MHz and provides an effective number of bits (ENOBs) of 6.8 when utilizing an oversampling ratio of 128×. The configured pipeline ADC provides an ENOB of 3.7 for a sampling frequency of 15 MHz.**

***Index Terms*—Field-programmable analog array (FPAA), reconfigurability, synthesis algorithm.**

I. Introduction

THE field-programmable analog array (FPAA) has attracted growing interest due to the increasing complexity and costs associated with fabricating and verifying prototype ICs in submicrometer technologies [1], [2], [3]. Similar to field-programmable gate arrays (FPGAs), which have been widely adopted for various digital applications, research and development of FPAA topologies has successfully led to the implementations of reconfigurable analog and mixed-signal circuits. A historical overview of the evolution of the FPAA is provided in [1].

An FPAA that utilizes floating-gate (FG) devices and a fully connected crossbar network is proposed that allows for the implementation of complex analog circuits including an AM receiver and an analog speech processor [4]. In [5], an FG FPAA with a routing architecture that includes a fully connected crossbar switch matrix is proposed to improve the density of functional blocks. An FG FPAA [6] with a Manhattan routing architecture is developed that permits prototyping of mixed-signal circuits including a voltage-controlled oscillator, an analog-to-digital converter (ADC), and a second-order sigma-delta modulator. In [7], a microprocessor is integrated with an FG FPAA to enable rapid reconfigurable mixed-signal computation. In addition to coarse-grain components such as OTAs, fine-grain components including transistors and capacitors are also provided in FG-based FPAAs [4], [5], [6], [7]. Therefore, FG FPAAs allow for both fine-grain and course-grain circuit configurations [4], [5], [6], [7]. A vector-matrix multiplier (VMM) combined with a winner-take-all (WTA) classifier is demonstrated on an FPAA platform in [8]. In [9], physical computation of real value problems is demonstrated on an FPAA. A switch-less FPAA fabric is developed in [10] and [11] that provides an improved frequency response when configured as a bandpass filter. In [12], analog functions are programmed on time-domain CABs that utilize the frequency and phase of digital pulses to transmit analog information. In addition, FPAA topologies are proposed that feature distinct CAB architectures that target specific applications [11], [12], [13], [14], [15], [16]. Tailored CAB topologies leverage circuit hierarchy by utilizing subcircuits that include operational transconductance amplifiers (OTAs) [11], [17] or switched capacitors [14], while allowing for transistor-level configuration [15], [18].

The tradeoff between the programmability and the performance of an FPAA is dependent on the smallest programmable element within the configurable analog block (CAB) and the routing strategy used to connect the fundamental elements. Utilizing transistors as the smallest programmable element provides the highest programmability. The routing switches are utilized to connect transistors within the FPAA fabric and implement various analog circuits. Alternatively, small analog circuit blocks that include operational amplifiers, comparators, and integrators serve as the smallest programmable components within a CAB, with routing switches connecting the small analog circuit blocks within the FPAA fabric to implement larger circuits. The utilization of fixed small analog circuits improves performance by minimizing the parasitic impedance introduced by routing switches placed between transistors, which degrade operating frequency and bandwidth. However, utilizing small analog circuits as the smallest programmable component constrains the FPAA to specific families of circuits, which limits the types of analog circuits implemented. Routing strategies also impact programmability and performance. Switch-based routing networks provide greater flexibility as compared to fixed interconnects. In [10], fixed interconnects are used to connect different CABs, which increases the circuit bandwidth but also limits the types of circuit topologies that are implemented. In [16], a hybrid

Received 15 November 2024; revised 7 February 2025; accepted 25 February 2025. Date of publication 29 May 2025; Date of current version 1 July 2025. This work was supported in part by the National Science Foundation under Grant CNS-1751032 and in part by the Office of Naval Research under Award N00014-22-1-2071. *(Corresponding author: Ziyi Chen.)*

The authors are with the Electrical and Computer Engineering Department, Drexel University, Philadelphia, PA 19104 USA (e-mail: zc372@drexel.edu).

Digital Object Identifier 10.1109/TVLSI.2025.3553538

routing strategy is utilized, combining fixed interconnects and switches. The routing switches are included within the CAB while fixed interconnects are used between CABs, which addresses the bandwidth limitations of large analog circuits but restricts the number and type of circuits that are implemented.

In this work, an FPAA architecture is introduced that allows for the implementation of reconfigurable AMS circuits. The developed CAB is comprised of programmable components that include transistor pairs (TPs), capacitors, and resistors, which enable both transistor-level programmability and the configuration of passive devices. While a single CAB is sufficient to implement small analog circuits, complex analog topologies are configured on multiple CABs. Consequently, the proposed architecture not only supports transistor-level configuration but also facilitates the implementation of coarse-grain system-level circuits. To ease the design and implementation of analog circuits on the FPAA, a synthesis flow is developed that automatically places and routes an analog circuit onto the CABs of the FPAA fabric. A novel algorithm is developed that maps a given netlist onto the FPAA fabric, where the vertical and horizontal location of each transistor of an analog circuit is algorithmically determined. In addition, an algorithm that utilizes circuit isomorphism is developed that matches TP topologies to four baseline configurations. For the routing of interconnects, a Dijkstra algorithm is utilized to find the shortest path between two nodes while avoiding collision. The primary contributions of the article are as follows.

1) An FPAA architecture that efficiently combines transistor-level configuration and coarse-grain system-level configuration.
2) The development of a placement algorithm that efficiently maps a given analog circuit onto the FPAA fabric, where the vertical and horizontal location of each transistor is determined and assigned onto a CAB.
3) The development of a technique that matches the topology of a TP to a set of predefined topologies provided in a library. Once matched, routing switches within the programmable TPs are utilized to implement the determined topology.

The rest of the article is organized as follows. Prior work describing implementations of FPAAs is presented in Section II. The FPAA architecture developed as part of this work is described in Section III. The synthesis methodology and algorithms that map an analog netlist onto the FPAA CAB are presented in Section IV. The characterization of small analog circuits implemented on the FPAA is provided in Section V. The configuration of more complex analog circuits is described in Section VI. A comparison of the proposed FPAA architecture with prior FPAA implementations is provided in Section VII. Use cases of the proposed FPAA are provided in Section VIII. Some concluding remarks are provided in Section IX.

II. Prior Work

The development of programmable analog circuits lags behind that of programmable digital circuits, with only a few notable efforts contributing to FPAA technology. In one such effort, a programmable analog device array (PANDA) is developed that enables transistor-level configuration [15]. The PANDA cell, which contains either one or three programmable transistors, emulates the behavior of transistors of varying widths and lengths. To ensure correct performance, a sizing algorithm [18] matches the current, output resistance, and transconductance of the PANDA cell with that of the target transistor. Switches implemented as transmission gates (TGs) are utilized to establish connections between PANDA cells.

A switch-less FPAA is described in [10], which is utilized to implement tunable OTAs. Each CAB includes seven transconductive cells that are digitally tunable, which are configured based on the parallel connection of six programmable OTAs.

FG FPAAs, as described in [4], have been used to implement more complex analog circuits. In such FPAAs, FG devices are programmed with high precision through hot-electron injection, while the charge on the FG node is erased through electron tunneling [19], [20]. Each CAB is comprised of coarse-grained elements that include OTAs, FG current mirrors, and Gilbert muxes. In addition, passive components including resistors and capacitors are also provided. FG switches implemented as FG transistors are utilized as routing components, and the OTAs are biased also using FG transistors. The work presented in [6] introduces an FPAA topology for mixed-signal applications. In addition to the CAB, configurable logic blocks (CLBs) are utilized to implement digital circuits. The elements in the CLB include look-up tables (LUTs) and JK flip-flops. An FG-based mixed-signal array is proposed in [7]. A microprocessor is utilized to allow for rapid configuration of the routing fabric. Circuits that include a ladder-filter-based, linear phase analog filter [21], a hopf bifurcation element based on transconductance amplifiers [22], and an adaptive-Q bandpass filter [23] are demonstrated on the FG FPAA. A programmable analog standard cell library is proposed based on the abstraction of analog computational elements that are programmable on the FG FPAA fabric [24]. The analog standard cells provide a broader design capability for analog computing systems. An open-source toolset [3] is proposed that enables analog-digital-software co-design on FPAAs. The framework utilizes high-level software in Scilab/Xcos to convert the high-level block description to a file in Berkeley Logic Interchange Format (BLIF). The open-source tool VPR/VTR [1] is utilized to place and route the netlist onto an SoC FPAA fabric. In addition, the *vpr2swcs* algorithm generates the list of switches in the on state based on topological details provided in a file describing the architecture of the IC.

An FPAA described in [14] utilizes zero-crossing detectors to implement analog circuits on a reconfigurable fabric. Each CAB contains a zero-crossing circuit [14] and a switch-capacitor network, which allows for the implementation of active ladder filters and pipeline ADCs. However, zero-crossing circuits are utilized for specific analog functions and are not suitable for implementing generic analog circuits.

A time-domain FPAA is described in [12] that emulates analog functions using digital circuitry. The time-domain CAB is utilized to process digital pulses with modulated frequency and phase. Circuits implemented on a single CAB include time-to-digital converters, phase interpolators, digital pulsewidth modulators, and digitally controlled oscillators. In addition, CLBs and arithmetic logic units (ALUs) are added to the programmable fabric to allow for the configuration of mixed-signal circuits, including phase-locked loops (PLLs) and ADCs.

III. Architecture of the FPAA

Manhattan routing is widely used in FPAAs [1]. In this work, two-level Manhattan routing is utilized to enable connections between the CABs and between transistors within the

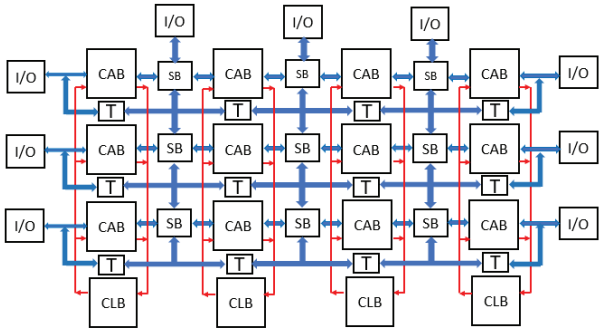


Fig. 1. Architecture of the FPAA fabric that includes a 3×4 CAB array and a 3×4 SB array. One CLB is added to each column of the CAB matrix for the implementation of digital circuitry.

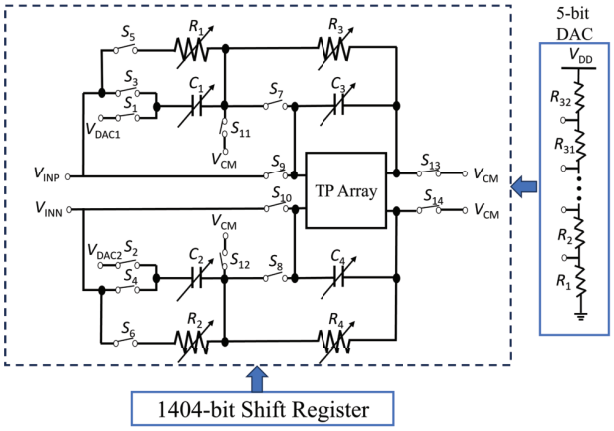


Fig. 2. Schematic representation of a CAB. The TP array allows for transistor-level configuration. Resistors R_1 to R_4 provide a programmable resistance, while capacitors C_1 to C_4 provide a programmable capacitance. The switches S_1 to S_{14} of each CAB are programmed by data stored in the shift register.

CAB. The architecture of the proposed FPAA features a 3×4 CAB matrix, as shown in Fig. 1. The switch boxes (SBs) allow for global connections between the CABs. Transmission gates (TGs) are utilized as routing switches that connect SBs of adjacent columns. A programmable CLB is included in each column of the CAB matrix, which enables the implementation of configurable analog and mixed-signal circuits.

A. Architecture of the Configurable Analog Block (CAB)

The CAB, which allows for the programming of various analog circuits of mid-complexity, is a subcomponent of the FPAA. In this article, the proposed CAB is programmed into various continuous-time analog circuits and switch-capacitor circuits that operate in the discrete-time domain. The architecture of the CAB is shown in Fig. 2, which is comprised of a transistor pair (TP) array, four programmable transistors, four programmable capacitors, and 11 routing switches implemented as TGs that allow access to both active and passive components. Shift registers are utilized to store the programmble bitstream. Each CAB includes a 1404-bit D-flip-flop shift register, which occupies an area of 0.034 mm^2. The shift registers of the FPAA include a total of 17 108 bits. The shift registers used to program the CABs within the same column of the CAB matrix are cascaded, enabling column-wise programming of the CABs of the FPAA. The bitstream is first loaded into the shift registers and then loaded to the TG switches of each CAB. The Anadigm FPAA [25] utilizes two switch matrices to route analog signals from a

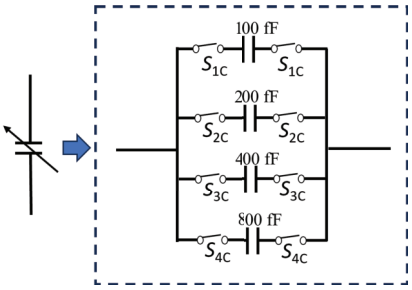


Fig. 3. Schematic representation of a programmable capacitor. The tunable range of the capacitance is from 100 fF to 1.5 pF.

bank of eight programmable capacitors or from the feedback paths of the two internal op amps and a single comparator. When implementing a switched-capacitor (SC) circuit, the input and feedback capacitors are selected from the bank of programmable capacitors through the switch matrices. In this work, to sample input signals, the CAB includes two programmable input capacitors that are placed between the input ports of the CAB and the input ports of the TP array as shown in Fig. 2. In addition, two programmable feedback capacitors are placed between the input and output ports of the TP array, which are utilized when implementing a switch-capacitor integrator. The programmable resistor is implemented as a set of subresistors of different values connected in parallel and activated by TG switches. Each programmable resistor, designated as R_1 to R_4 in Fig. 2, provides a tunable resistance of 1 kΩ to 5 kΩ with a unit step size of 1 kΩ. The programmable capacitor is implemented by connecting binary weighted capacitances in parallel as shown in Fig. 3. A minimum capacitance of 100 fF is provided when the switches S_{1C} are in the active ("ON") state and the switches S_{2C}, S_{3C}, and S_{4C} are in the OFF state. A maximum capacitance of 1.5 pF is provided when switches S_{1C}, S_{2C}, S_{3C}, and S_{4C} are all set to the ON state. Each programmable capacitor, designated as C_1 to C_4 in Fig. 2, provides a tunable capacitance of 100 fF to 1.5 pF with a unit step size of 100 fF. The passive components allow for the implementation of various analog circuits, including continuous-time integrators, switch-capacitor integrators, and multiply digital-to-analog converters (MDACs). In addition, a 5-bit digital-to-analog converter (DAC) based on a resistor ladder topology, as shown in Fig. 2, is included in each CAB. The DAC provides a dc bias voltage to each transistor of the TP array. The 5-bit resistor-ladder DAC consists of 32 resistors, each with a value of 50 Ω. The programmable bias voltage provided by the DAC is in the range of 0 V to 1.2 V, with a step size of 37.5 mV. The flexibility of the CAB, therefore, allows for the configuration of both fine-grain and coarse-grain circuits, which makes the CAB a versatile circuit block capable of implementing a diverse set of analog functions.

B. Architecture of the Transistor Pair (TP) Array

Allowing for fine-grain configuration [2], [26] provides an improved tradeoff between the efficiency and the flexibility of the analog circuits implemented on the FPAA fabric. The fine-grain FPAA architectures described in [4], [5], [6], and [7] allow switches to operate between the lowest level of components, i.e., individual transistors, to maximize the configurability and flexibility of the programmable fabric [2], [26]. In this work, the programmable TP is the lowest level component provided within the TP array. The transistor-level programmability of the TP array allows for the configuration

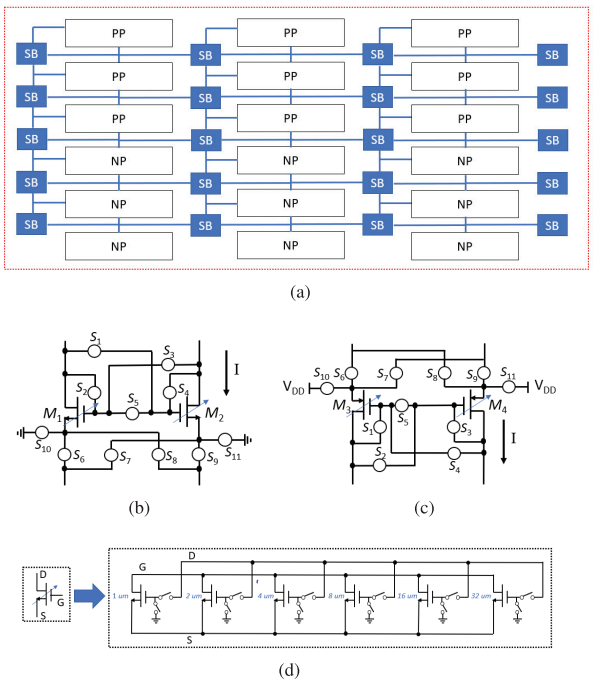


Fig. 4. Block and schematic representations of the (a) architecture of the TP array that includes a 6 × 3 matrix of TPs, (b) structure of an NMOS TP NP, (c) structure of a PMOS TP PP, and (d) structure of a reconfigurable transistor with a tunable width of 1 to 63 μm. The PMOS and NMOS TPs utilize switches S_1 to S_{11} for topology configuration (current direction is also shown).

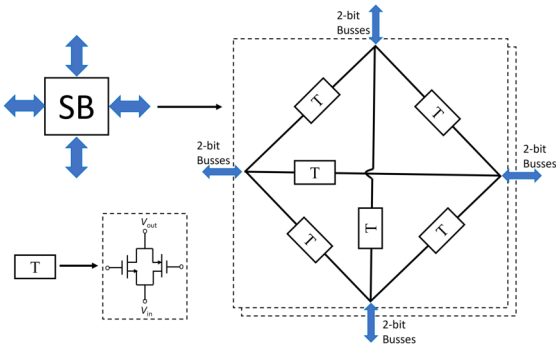


Fig. 5. Schematic representation of a switch box (SB). The "T" represents a TG switch.

of complex analog circuits. The routing switches between the TPs are utilized to form connections. A programmable TP is configured into four different topologies that include a current mirror, a regeneration loop, a current source pair, and a diode-connected pair. Utilizing the TP as a basic routing element and the programmability provided by a TP simplifies the process of mapping a differential analog circuit topology onto the FPAA fabric. The TP array is composed of a 6 × 3 matrix of TPs, as shown in Fig. 4(a). The matrix includes PMOS TPs (PP), NMOS TPs (NP), and SBs. The transistor-level configuration of the TP array allows for the fine-tuning and customization of individual CABs to meet specific analog circuit requirements, providing an alternative to established coarse-grain FPAA topologies [27].

Each SB contains six routing switches that form paths between any two or more directions. The TP array is programmed into an analog circuit, with circuit parameters further tuned by adjusting the programmable TPs. The topology of an NP and PP is shown in Fig. 4(b) and (c), respectively. Switches S_1 to S_5 are used to program the two NMOS or PMOS TPs into a current mirror, a regeneration loop, a current source pair, or a diode-connected pair. Switches S_6 and S_7 are utilized to merge the dc current from the two transistors into the left current path, while switches S_8 and S_9 perform the same function for the right current path. Switches S_6 to S_9 are also programmed as isolators if required, preventing the flow of current through the drain of the TPs. Switches S_{10} and S_{11} connect the source of the PMOS transistors to V_{DD} (the positive power supply voltage) and the source of the NMOS transistors to GND (the ground reference voltage). The proposed switch configuration ensures that the V_{DD} and GND voltage sources are universally available to all transistors of the TP array, which provides increased flexibility when designing an analog circuit.

The NP and PP TPs (M_1, M_2 and M_3, M_4 shown in Fig. 4(b) and (c), respectively) provide programmable widths, where six binary weighted transistor widths are connected in parallel as depicted in Fig. 4(d). The programmable transistors provide a minimum width of 1 μm and a maximum width of 63 μm with a step size of 1 μm. The tuning range of the transistor width provided by the proposed TP array surpasses that of coarse grain CABs, which enables precise control over circuit parameters including the gain and bandwidth of an operational amplifier (op amp). The high level of programmability provided by the TP array allows for the optimization of the performance of an analog circuit to meet specific application requirements with enhanced precision.

C. Routing Network

The routing network of the FPAA is depicted in Fig. 1, where each CAB is connected to adjacent CABs through SBs. The SBs utilize TGs as routing switches. To reduce the ON resistance and the corresponding dc voltage drop across the routing switches, the NMOS transistors are set to a width of 30 μm and a length of 60 nm, while the PMOS transistors are set to a width of 60 μm and a length of 60 nm. As a result, the ON resistance of a routing switch is less than 50 Ω, and the parasitic capacitance is approximately 21 fF. The schematic of the implemented SB is shown in Fig. 5, which is similar to the SB described in [6]. The "T" represents a TG switch. Six TG routing switches form connections between the four routing directions. Each SB utilizes 12 routing switches and four 2-bit buses to allow for differential connections between the TPs. The parasitic capacitance seen from each input port of the SB is approximately 70 fF. The proposed routing network ensures efficient signal routing and minimal signal degradation within the FPAA.

D. CLB Architecture

The FPAA includes CABs along with CLBs, which enable the implementation of mixed-signal circuits, including pipeline ADCs and delta-sigma ADCs. For the pipeline ADC, the CLB implements a decoder, while for the delta-sigma ADC, the CLB is programmed to implement an SR latch. Each CLB is comprised of five two-input lookup tables (LUTs), as shown in Fig. 6. The three inputs, V_{in1}, V_{in2}, and V_{in3}, are sourced from the analog output of a CAB located in the same column. The five voltage references V_{out1}, V_{out2}, V_{out3}, V_{out4}, and V_{out5} are derived from the outputs of the five two-input

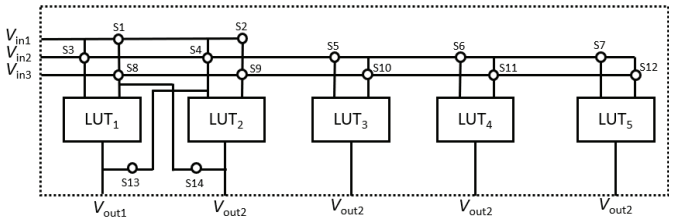


Fig. 6. Architecture of a CLB includes five two-input LUTs, three voltage inputs, and five voltage outputs.

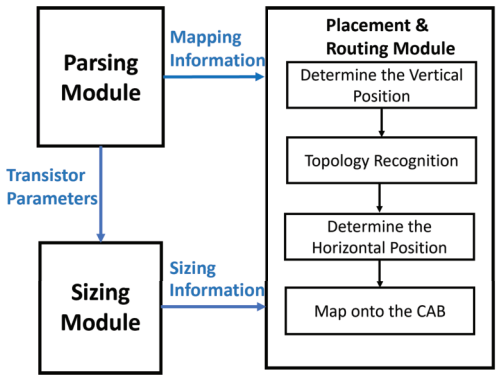


Fig. 7. CAD framework that includes netlist parsing, transistor sizing, and placement and routing modules.

lookup tables, which are programmed into combinational logic circuits. For the implementation of a pipeline ADC, LUT$_1$, LUT$_2$, and LUT$_3$ are configured into an OR gate, an XOR gate, and an AND gate, respectively. The voltages V_{out1}, V_{out2}, and V_{out3} represent the one-hot code generated by the 1.5-bit ADC within each pipeline stage. The one-hot code is used to program the DAC and generate the reference voltage for the ADC. In addition, LUT$_4$ is programmed as an AND gate and LUT$_5$ as an XOR gate. The voltages V_{out4} and V_{out5} represent the quantized digital outputs of each pipeline stage. When implementing a delta-sigma modulator, LUT$_1$ and LUT$_2$ are programmed as NOR gates. Switches S_{13} and S_{14} of the CAB are turned on to configure LUT$_1$ and LUT$_2$ into an SR latch. The voltages V_{out1} and V_{out2}, which represent the output of the SR latch, are used to program the DAC and generate the reference voltage for the SC integrator. The combination of configurable analog and digital blocks provides a highly versatile and adaptive platform that allows for the implementation of complex mixed-signal circuits, which enables the use of the FPAA in a wide range of analog and digital signal processing applications.

IV. CAD Methodologies to Synthesize Analog Circuits

Algorithms developed to automate the synthesis of analog circuits on the FPAA fabric are described in this section. The proposed CAD flow consists of a netlist parsing module, a transistor sizing module, and a placement and routing module, as depicted in Fig. 7. The algorithms support two flows to configure the FPAA: one flow implements a structural netlist onto the FPAA fabric, while the other enables direct programming of an analog circuit onto the FPAA fabric. The parsing module takes a circuit SPICE netlist as input and extracts essential transistor-level information, including the transistor width, transistor length, and the net name connected to the gate, drain, and source terminals of each transistor. The parsed information is utilized to determine the relative vertical and horizontal location [27] of each transistor of the netlist and to ensure the proper mapping of the transistors onto the TP arrays of the FPAA. The sizing module utilizes a dc nodal analysis of the target analog circuit to adjust the width of the transistors of the FPAA fabric. The goal is to match the circuit parameters of the FPAA, including the dc current, to that of the circuit defined by the netlist. The resizing of the devices guarantees that the analog circuit implemented on the FPAA behaves according to the desired specifications.

Algorithm 1 Pseudocode to Determine the Vertical Level of Transistors of a Circuit

```
Input: circuit netlist φ
Output: list of vertical locations L_ver
mosVec = getTransistorInfo(φ)
netsVecVec[0].append("VDD")
for i from 0 to len(mosVec) do
    vertiLevel = i+1
    for j from 0 to len(mosVec) do
        if isNMOS(mosVec[j]) and
         find(mosVec[j].drain, netsVecVec[i]) then
            mosVec[j].vertiLevel = vertiLevel
            netsVecVec[i+1].append(mosVec[j].source)
        else if isPMOS(mosVec[j]) and
         find(mosVec[j].source, netsVec) then
            mosVec[j].vertiLevel = vertiLevel
            netsVecVec[i+1].append(mosVec[j].drain)
    return mosVec
```

A. Determination of Transistor Vertical Level

Considering that analog circuits include a dc current path from V_{DD} to GND, the vertical level of each transistor of the circuit is determined by the relative position of each device with respect to the power rail (V_{DD}) and the ground rail (GND). For the TP array, only PMOS transistors are connected to the V_{DD} rail, while only NMOS transistors are connected to the GND rail. The pseudocode to determine the vertical level of each transistor is provided as Algorithm 1.

The algorithm first parses a circuit netlist, with information extracted on the vertical level of each transistor. A 2-D vector, *netsVecVec*, is assigned the vertical level of circuit nets in one dimension, and a list of all nets within that vertical level as the second dimension. The PMOS transistors with the source terminal connected to V_{DD} are assigned to level 1. The nets connected to the drain terminal of level 1 PMOS transistors are also assigned to level 1. An NMOS transistor is labeled as level $i+1$ if the drain of the transistor is connected to level i nets, and a PMOS transistor is labeled as level $i+1$ if the source is connected to level i nets.

The current path from V_{DD} to GND determines the dc voltage of internal nodes. If the current flows through an NMOS transistor, the dc current from the drain to the source establishes the dc operating point of the transistor and creates a voltage drop between the drain and source terminals. Similarly, in a PMOS transistor, the current flow produces a voltage drop between the source and drain terminals. In addition, transmission-gate switches are used to connect the drain and source terminals of different transistors within the TP array, which results in a voltage drop across the switches due to

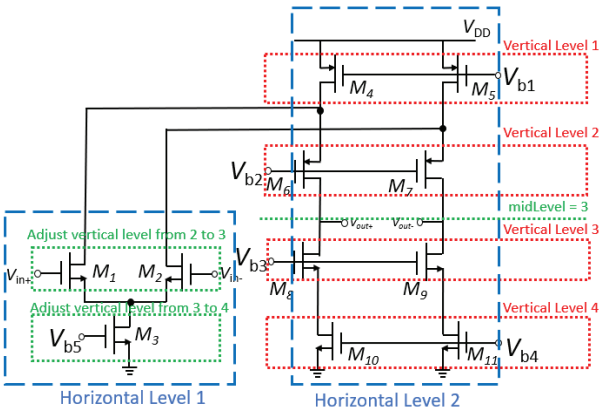


Fig. 8. Circuit schematic of a folded-cascode op amp. The circuit netlist is mapped to TPs on vertical levels 1 to 4, and horizontal levels 1 and 2.

the ON-resistance of the switches. The topology of the CAB constrains the maximum vertical level of a target circuit to six. The six vertical levels are sufficient to cover most analog circuits, while considering that each transistor on the current path from V_{DD} to GND contributes to a dc voltage drop that reduces the available voltage headroom across the vertical stack of transistors. Therefore, the assignment of the vertical level ensures efficient utilization of the available resources and the optimal performance of an analog circuit synthesized onto the FPAA fabric. The placement module takes the netlist of an analog circuit as input. The netlist is then parsed and the vertical position of the transistor is determined. For the transistors in each vertical level, a topology recognition algorithm is executed that determines the topology that each TP is to be set to. The horizontal position of the TP is then determined. Finally, the TP is then explicitly mapped onto the CAB fabric according to the determined vertical and horizontal positions.

B. Adjustments to the Assigned Vertical Level

When initially determining the vertical level of each transistor of a folded-cascode op amp, which is shown in Fig. 8, the NMOS transistors M_1 and M_2 are assigned to level 2, while the PMOS transistors M_6 and M_7 are assigned to level 3. However, due to the topology of the TP array, where the first three rows consist of PMOS transistors and the last three rows consist of NMOS transistors, the PMOS and NMOS transistors are not properly assigned to the correct vertical level. To address the error in the assignment of the vertical level, an algorithm is developed that modifies the vertical level of the mapped transistors to match the topological structure of the circuit, with pseudocode of the routine provided as Algorithm 2.

The input to the algorithm is the profile of the transistors provided by the initial parsing of the netlist, which includes information on the vertical position. The algorithm begins by grouping all nets connected to PMOS transistors into *Pnets*, and all nets connected to NMOS transistors into *Nnets*. The middle level of nets are identified, which are the nets with connections to both NMOS and PMOS transistors, and are defined by *midLevel* in Algorithm 2. The parameter *RefineNets* stores the nets connected to the source of a transistor with a modified level, while *startLevel* represents the lowest level of nets that connect to both NMOS and PMOS transistors.

Next, the algorithm iterates through all NMOS transistors and identifies those whose vertical level is less than *midLevel*. The vertical level of transistors less than *midLevel* is increased

Algorithm 2 Pseudocode to Modify the Vertical Level of Transistors of a Circuit

```
Input: mosVec, netsVecVec, mosLevel
Output: mosVec with updated vertical location
maxLevel = FindMaxVert(mosVec)
Pnets = NetsToPMOS(mosVec)
Nnets = NetsToNMOS(mosVec)
midLevel = 0
startLevel = 0
for i from 1 to maxLevel do
    for net in netsVecVec[i] do
        if find(Pnets, net) and find(Nnets, net) then
            midLevel = i
for i from maxLevel to 1 do
    for net in netsVecVec[i] do
        if find(Pnets, net) and find(Nnets, net) then
            startLevel = i
RefineNets = []
for nmos in mosLevel[startLevel] do
    if nmos.VertiLevel < midLevel then
        nmos.VertiLevel += midLevel -
         nmos.VertiLevel
        RefineNets.append(nmos.source)
SearchLevel = midLevel+1
while GND ∉ RefineNets do
    for nmos in mosVec do
        if nmos.drain ∈ RefineNets && nmos.vertiLevel
         == SearchLevel then
            RefineNets.append(nmos.source)
            nmos.VertiLevel += midLevel -
             nmos.VertiLevel
    SearchLevel++
```

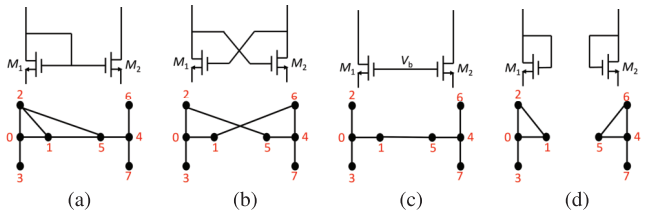


Fig. 9. Graph representation of a (a) current mirror, (b) regeneration loop, (c) current source pair, and (d) diode-connected pair. The same topologies are programmable on both NMOS and PMOS TPs.

to allow for the placement of such transistors into the last three rows of the TP array. The algorithm continues until the GND net is included in *RefineNets*, which indicates that all the transistors of the circuit have been updated with the appropriate level number. By executing the algorithm, the vertical levels of the NMOS and PMOS transistors are adjusted to ensure proper vertical placement in the CAB, as shown for the folded-cascode op amp in Fig. 8.

C. TP Matching

The topology of TPs of the same vertical level is matched to four predefined configurations: a current mirror, a regeneration loop, a current source pair, and a diode-connected pair. The topology of a TP is mapped to a graph of the four predefined configurations, as depicted in Fig. 9, where each transistor is

represented by four nodes. For the regeneration loop shown in Fig. 9(b), eight nodes comprise the graph, where nodes 0, 1, 2, and 3 represent transistor M_1, and nodes 4, 5, 6, and 7 represent transistor M_2. The net between the drain of M_1 and the gate of M_2 is denoted by the edge between nodes 2 and 5, while the net between the drain of M_2 and the gate of M_1 is represented by the edge between nodes 1 and 6.

After parsing the netlist and determining the vertical level of each transistor, the matching algorithm performs a priority search of the TPs of each vertical level, providing the highest priority to the identification of regeneration loops, followed by current mirrors, diode-connected pairs, and finally, current sources. The comparison is limited to transistors on the same vertical level, which reduces the number of isomorphic topologies to search as multilevel topologies including cascodes are not accounted for.

Algorithm 3 Pseudocode to Determine the Horizontal Level of Transistors of a Circuit

```
Input: mosVec
Output: mosVec with updated horizontal location
maxLevel = FindMaxVert(mosVec)
mosLevel = [NULL]*maxLevel
for i from 0 to len(mosVec) do
    mosLevel[mosVec[i].vertiLevel].append(mosVec[i])
leftNets = []
rightNets = []
occupy = [0,0,0]
for i from 0 to MaxLevel do
    for Config in TranPairConfigurations do
        for mos in mosLevel[i] && findConfig(mos) do
            if Occupy[2]==0 then
                mos.horiLevel = 2
                leftNets.append(mos.drain)
                rightNets.append(mos.pair.drain)
            else if Occupy[1]==0 then
                mos.horiLevel = 1
                leftNets.append(mos.drain)
                rightNets.append(mos.pair.drain)
            else
                mos.horiLevel = 3
                leftNets.append(mos.drain)
                rightNets.append(mos.pair.drain)
```

D. Determination of Transistor Horizontal Level

After determining the vertical level of each transistor, the parsing module proceeds to determine the horizontal level. With both the vertical and horizontal levels of each transistor determined, the parsing module provides the necessary data needed by the place and route module to accurately map the transistors onto the FPAA fabric. The pseudocode describing the routine that determines the horizontal location of each transistor is provided as Algorithm 3. The horizontal level is considered for TPs that share the same vertical level, where the parameter *mosLevel[i]* stores all TPs of the vertical level *i*.

For each vertical level, the algorithm compares the TPs to graphs stored in a library. The TPs are placed in a prioritized order, with regeneration loops placed first, followed by current mirrors, then current sources, and finally TPs with differential inputs. A selected transistor pair is initially placed in the middle column of a given row of the TP array, followed by placement in the first column, and finally in the last column of the same row. The algorithm ensures the efficient and optimal placement of the TPs, while accounting for the specific requirements of the target circuit topology and also maximizing the use of the available resources of the FPAA fabric.

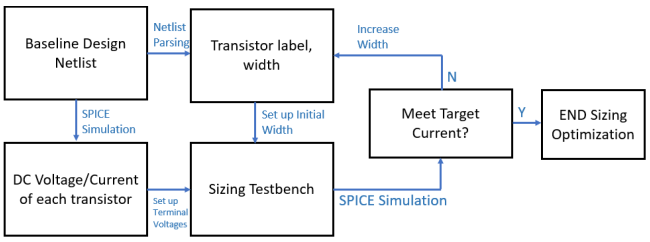


Fig. 10. Block representation of the transistor sizing flow.

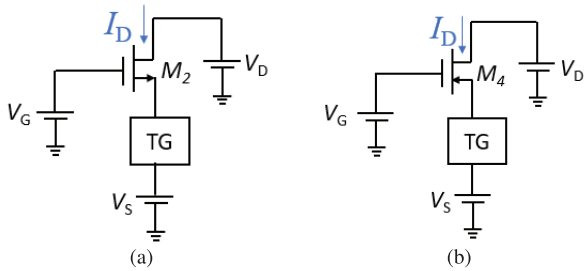


Fig. 11. Circuit topologies utilized to (a) apply the determined dc voltages and tune the NMOS transistor width while maintaining the drain current I_D and (b) apply the determined dc voltages and tune the PMOS transistor width while maintaining the drain current I_D.

E. Transistor Resizing

Unlike in FPGAs, where the routing switches are connected to the gates of transistors, the routing switches of an FPAA carry current, which leads to voltage drops across the switches. Mapping the target transistors onto the FPAA with the originally defined sizes of the netlist results in a deviation in the dc operating point of the transistors from the ideal value. To address the shift in dc operating point, a resizing module is developed that considers the ON-resistance of the routing switches.

A block diagram that depicts the stages of the transistor sizing flow is shown in Fig. 10. First, the dc operating point of each transistor is extracted from SPICE simulation. Then, a routing switch is added along the path carrying the current of the transistor, as shown for the NMOS transistor and PMOS transistor in Fig. 11(a) and (b), respectively. The objective for a given gate, drain, and source voltage is to match the dc current flowing through the transistor and TG routing switch to the ideal expected current of the transistor by modifying the transistor width. The proposed programmable TP topology introduces only a single routing switch in the dc current path, which increases the voltage headroom as compared to the PANDA FPAA [15]. In addition, the effect on the dc current due to the routing switches within the SBs is accounted for when in the ON-state.

The algorithm for resizing is executed after determining the vertical and horizontal levels of each transistor. The resizing module ensures that the analog circuits implemented

on the FPAA maintain the desired dc operating point, while mitigating the effect of the ON-resistance of the routing switches on the performance and accuracy of the circuit. By considering the parasitic effects of the switches and paths during the synthesis of the analog circuit, the proposed CAD flow improves the optimal performance and the precision of the circuit parameters provided by the FPAA.

F. Routing Algorithm

Once the vertical and horizontal locations of each transistor are determined, the transistors are mapped onto the TP array. The Manhattan routing network within the TP array is represented as an undirected graph, where the vertices represent the routing switches and the edges represent the interconnects between the switches. For routing, a Dijkstra algorithm [28] is executed that finds the shortest path between two nodes, which implicitly indicates the routing switches within the CAB to program in the ON state. Due to the architecture of the TP array, implementing a differential topology on the proposed FPAA is equivalent to mapping TPs from a target circuit netlist onto the programmable TPs of the FPAA array. An Ocean script containing the programming bitstream is then written for SPICE simulation. The routing algorithm does not account for the capacitance of the path. However, the algorithm ensures that the routing path traverses the minimum number of routing switches. Note that once the vertical and horizontal locations of the TPs are determined, the source-drain connections between PMOS and NMOS TPs utilize a fixed interconnect, which increases the voltage headroom of the implemented analog circuit by reducing the number of switches in the dc current path.

V. Characterization of Programmed Analog Circuits

The characterization of configured analog circuits is described in this section. The transistor-level configuration is provided through the TP array included in each CAB, while the utilization of multiple CABs enables the implementation of complex analog circuits.

A. Folded-Cascode Op Amp

Op amps serve as fundamental building blocks in AMS systems, including as components for receivers, ADCs, and PLLs. The folded-cascode op amp [29] is widely favored as the gain provided is greater than that of a single-stage op amp and the voltage headroom provided is greater than that of a telescopic op amp.

The schematic of a folded-cascode op amp is shown in Fig. 8. The three auto-mapping and sizing algorithms described in Section IV are used to program the folded-cascode op amp onto the FPAA fabric. After executing the algorithm that determines the vertical level of each component (only transistors in this case), which is described by Algorithm 1, transistors M_1 and M_5 are labeled as part of level 1, transistors M_1, M_2, M_6, and M_7 are labeled as part of level 2, transistors M_3, M_8, and M_9 as part of level 3, and transistors M_{10} and M_{11} as part of level 4. Note that the mapping of NMOS and PMOS transistors onto the same row of the reconfigurable TP matrix is not permitted.

After executing the algorithm that modifies the vertical level of the transistors, as described by the pseudocode provided as

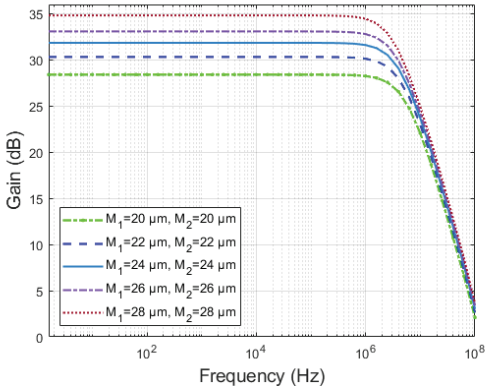


Fig. 12. AC response of a programmed folded-cascode op amp highlighting a tunable gain and bandwidth.

Algorithm 2, transistors M_1 and M_2 are relabeled as level 3, while transistor M_3 is relabeled as level 4. The transistors in each vertical level are then compared and matched to TPs of a predifined library of topologies. TPs that are classified as current sources include M_4 and M_5, M_6 and M_7, M_8 and M_9, and M_{10} and M_{11}. In addition, transistors M_1 and M_2 are identified as a differential input pair, while M_3 is identified as a single-ended current source. Finally, the horizontal level of each transistor is determined by executing Algorithm 3. Transistors M_1, M_2, and M_3 are placed in horizontal level 1, while transistors M_4 to M_{11} are placed in horizontal level 2. A 5-bit resistor ladder DAC, as shown in Fig. 2, is utilized to generate the bias voltages V_{b1}, V_{b2}, V_{b3}, V_{b4}, and V_{b5}. The 5-bit DAC offers a voltage range of 0 V to 1.2 V in increments of 37.5 mV. The transistors are resized to consider the voltage drop across the routing switches, while also accounting for the parasitic impedence due to the routing of a current path through a SB. The ac response of the programmed folded-cascode op amp is shown in Fig. 12. A gain of 28.3 dB and a 3-dB bandwidth of 5.3 MHz is achieved when the width of M_1 and M_2 is programmed to 20 µm. In addition, a gain of 34.8 dB and a 3-dB bandwidth of 3.3 MHz is achieved when the width of M_1 and M_2 is programmed to 28 µm. The power consumption of the configured folded-cascode op amp is 1.5 mW. The limitation on the dc gain is primarily due to the maximum tunable width of the programmable transistors and the fixed length of the transistor. In this work, the tunable width ranges from 1 to 63 µm, with a minimum transistor length of 60 nm applied to all transistors. Expanding the range of tunable widths and incorporating transistors with larger lengths are required to increase the tunable gain of the op amp.

B. Comparator

A comparator utilizing the StrongARM latch topology, which is shown in Fig. 13, has been implemented on the FPAA fabric. The StrongARM comparator finds extensive use in various types of ADCs, including the pipeline ADC, flash ADC, delta-sigma ADC, and successive-approximation (SAR) ADC. The operation of the StrongARM comparator is divided into two phases. During phase 1, when the clock signal CLK is low, transistors M_8 to M_{11} pull nodes A and B and the output nodes to V_{DD}. In phase 2, when CLK is high, transistors M_8 to M_{11} are turned off, and the two regeneration loops, implemented by transistors M_3 and M_4 and transistors M_6 and M_7, set the voltage at the output node based on the voltage

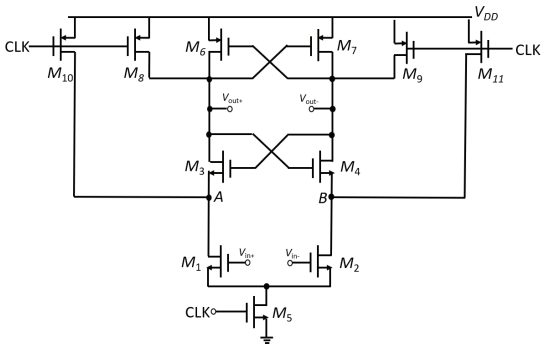


Fig. 13. Circuit schematic of a strongArm comparator.

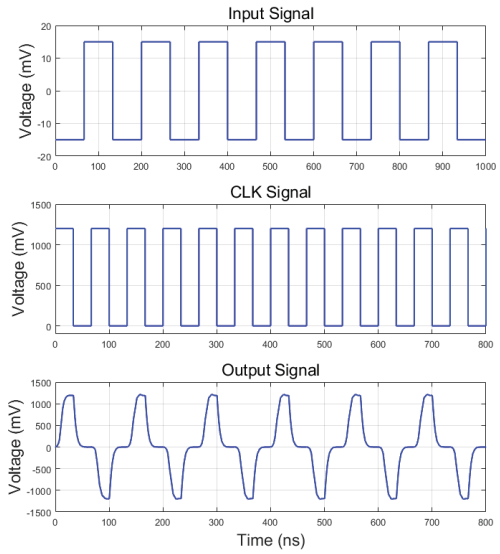


Fig. 14. Analysis of the transient response of a strongArm comparator, with the input signal, clock signal, and corresponding output signal characterized.

applied to the gate of the differential input pair formed by M_1 and M_2. Following the execution of Algorithm 1, transistors M_6 to M_{11} are labeled as belonging to vertical level 1. Level 2 includes transistors M_1 to M_4, while M_5 is labeled as a transistor connected to a *pnet* of vertical level 3. The vertical level of transistors M_1 and M_2 is relabelled as level 3 after executing Algorithm 2, while transistor M_5 is relabeled as level 4. Once all transistors are labeled with a vertical level, the topology of each TP in each vertical level is determined based on a search through a predefined library of topologies. The horizontal placement of each transistor on the TP array is then determined by executing Algorithm 3, with transistors M_8 and M_{10} placed in the first column, M_1 to M_7 in the second column, and M_9 and M_{11} in the third column of the CAB.

For the strongArm comparator, the output nodes are labeled as $V_{\text{out}+}$ and $V_{\text{out}-}$, as shown in Fig. 13. When the input voltage $V_{\text{in}+}$ is less than $V_{\text{in}-}$ and the clock signal (CLK) is high, the fall time corresponds to the time taken for $V_{\text{out}+}$ to drop from 90% to 10% of V_{DD}. When CLK is low, the output voltages $V_{\text{out}+}$ and $V_{\text{out}-}$ are reset to V_{DD}. The rise time corresponds to the time taken for $V_{\text{out}+}$ to increase from 10% to 90% of V_{DD}. The clock signal is applied to transistors M_5, M_8, M_9, M_{10}, and M_{11} through dedicated routing switches. The results from the transient simulation of a configured strongArm comparator are shown in Fig. 14. The offset voltage of the configured comparator is less than 15 mV, and the rise and fall times of the output are both less than 15 ns. The power consumption

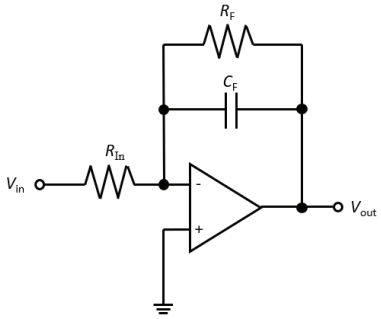


Fig. 15. Architecture of a continuous-time integrator.

of the configured comparator is 304.4 μW. The comparator is further utilized as a sub-block of a programmed pipeline ADC and a delta-sigma ADC, highlighting the versatility of the FPAA fabric in implementing hierarchical analog circuits of greater complexity.

C. Continuous-Time Integrator

A continuous-time (CT) integrator has been successfully implemented on the FPAA fabric. The schematic of the implemented CT integrator is shown in Fig. 15. The integrator consists of an op amp, an input resistor R_{In}, a feedback resistor R_F, and a feedback capacitor C_F. The feedback resistor R_F ensures that the dc operating point of the CT integrator is stable, while R_F and C_F set the cut-off frequency of the integrator. The unity-gain frequency of the integrator is determined by R_{In} and C_F, and is given by

$$f_{0\,\text{dB}} = \frac{1}{2\pi R_{\text{In}} C_F}. \tag{1}$$

A single CAB of the FPAA is utilized to implement the continuous-time integrator. A folded-cascode amplifier, as shown in Fig. 8, is configured through the TP array. Utilizing the device labels shown in Fig. 2, the input signal is applied through switches S_5 and S_6, which are connected to the programmable resistor array. In addition, switches S_7 and S_8 are in the ON state to connect to the programmable capacitor array, while the remaining switches are placed in the OFF state. The input resistance R_{In} is implemented using programmable resistors R_1 and R_2, each with a value of 1 kΩ. The resistance of the feedback path R_F is realized using programmable resistors R_3 and R_4, each with a value of 10 kΩ, while the feedback capacitor C_F is realized using programmable capacitors C_3 and C_4, each with a value of 500 fF.

The transient response of the configured continuous-time integrator is shown in Fig. 16. A square wave of 250 MHz frequency is applied as input to the integrator, while the output of the integrator is a triangular wave of equal frequency. The power consumption of the configured continuous-time integrator is 1.56 mW.

D. Discrete-Time Integrator

The proposed FPAA architecture not only supports continuous-time analog signal processing but also allows for discrete-time operation. The schematic of an implemented parasitic-insensitive switch-capacitor integrator is shown in Fig. 17, which includes an op amp, a sampling capacitor C_S, and a feedback capacitor C_F. The op amp of the integrator

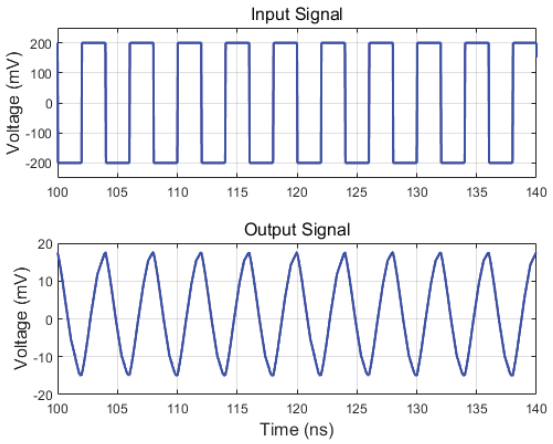


Fig. 16. Analysis of the transient response of a continuous-time integrator. The frequency of the input and output signal is 250 MHz.

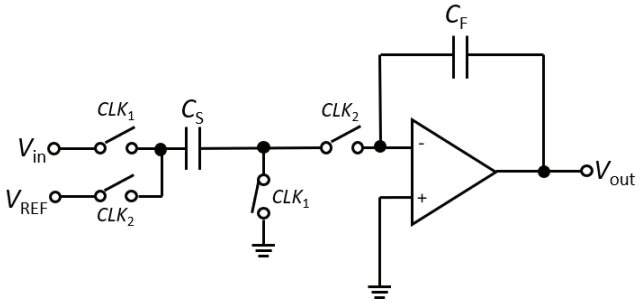


Fig. 17. Schematic representation of a switch-capacitor integrator.

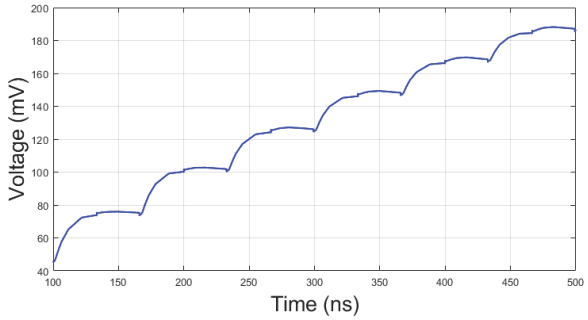


Fig. 18. Analysis of the transient response of an SC integrator. The step size of the integrator is 30 mV.

is implemented as a folded-cascode topology, as described in Section V-A, and is configured on the programmable TP array of a single CAB. The gain of the integrator is set by the ratio of the sampling capacitor to the feedback capacitor C_S/C_F.

Utilizing the device labels shown in Fig. 2, the differential sampling capacitor C_S is implemented using programmable capacitors C_1 and C_2, while the differential feedback capacitor C_F is implemented utilizing programmable capacitors C_3 and C_4. The gain of the integrator is set by the capacitance ratio C_1/C_3, which is the same as the capacitance ratio C_2/C_4. The programmable capacitors allow for flexibility when setting the gain. For example, to implement a unity-gain switch-capacitor integrator, the capacitance ratio C_1/C_3 is programmed to a value equal to 1. The operation of the programmed switch-capacitor integrator is divided into two phases. During phase 1, the input voltage is sampled by capacitors C_1 and C_2 through switches S_3 and S_{11} and switches S_4 and S_{12}, respectively. In phase 2, switches S_1, S_7, S_2, and S_8 are placed in the ON state,

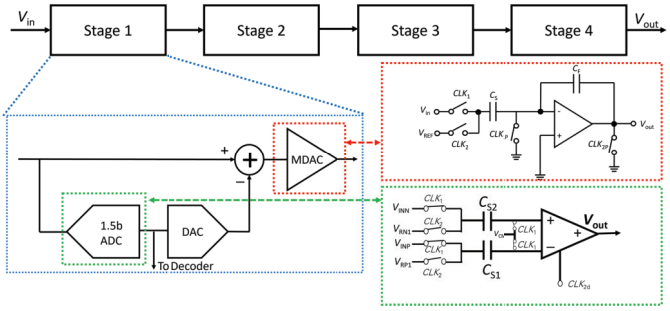


Fig. 19. Circuit representation of a 4-bit pipeline ADC. Each stage includes a sampling network and an MDAC.

while switches S_3, S_4, S_{11}, and S_{12} are placed in the OFF state. The charge stored on C_1 and C_2 is, therefore, transferred to C_3 and C_4, respectively, which results in an output voltage of $V_{\text{in}} \times (C_1/C_3)$. The transient analysis of the configured switch-capacitor integrator is shown in Fig. 18, which indicates a step size of 30 mV when integrating. The power consumption of the configured discrete-time integrator is 1.55 mW.

VI. Implementing Higher Complexity, Mixed Signal Circuits

Multi-CAB circuits are implemented on the FPAA, including a pipeline ADC and a delta-sigma modulator. The implemented mixed-signal circuits utilize both CABs and CLBs. For a programmed pipeline ADC, 12 CABs and four CLBs are required. For a programmed delta-sigma modulator, two CABs and one CLB are needed.

A. Pipeline ADC

The pipeline ADC [30] is widely used in applications that require a sampling frequency of 10 MHz to 100 MHz. In this work, a 4-bit pipeline ADC that features a multiplying DAC (MDAC) and a 1.5-bit sub-ADC are implemented as shown in Fig. 19. The MDAC, which includes a folded-cascode op amp, is programmed onto a single CAB. Within the CAB, the TP array is configured into an op amp, while the programmable capacitor arrays C_1 to C_4, which are shown in Fig. 2, implement the sampling and feedback capacitors.

The MDAC requires precise analog signal sampling when CLK_1 is high, which is achieved by activating switches S_3, S_{11}, S_2, and S_{12}, along with switches S_7, S_8, S_{13}, and S_{14} that reset the common-mode voltage of the op amp. A bottom-plate sampling is utilized, where the falling edge of CLK_{IP} precedes CLK_1. When CLK_2 is high, switches S_1 and S_2 are placed in the ON state, which enables the DAC to compute the output voltage through the transfer of charge between the sampling capacitors C_1 and C_2 and the feedback capacitors C_3 and C_4, respectively. The voltage at the output of the MDAC is given by

$$V_{\text{out}} = \frac{C_S}{C_F}\left(V_{\text{in}} - V_{\text{DAC}}\right), \tag{2}$$

where C_S represents the sampling capacitance, C_F the feedback capacitance, and V_{DAC} the output voltage from the DAC. The 1.5-bit sub-ADC employs dual strongArm comparators, the architecture of which is shown in Fig. 13. In addition, a sampling network is implemented with switch-capacitors and is integrated with each comparator. The voltages V_{INP} and

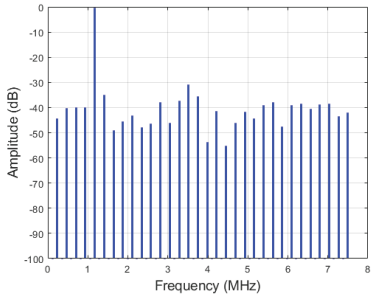


Fig. 20. Characterization of the FFT spectrum of the configured pipeline ADC. The frequency of the input signal is 1.17 MHz and the sampling frequency is 15 MHz.

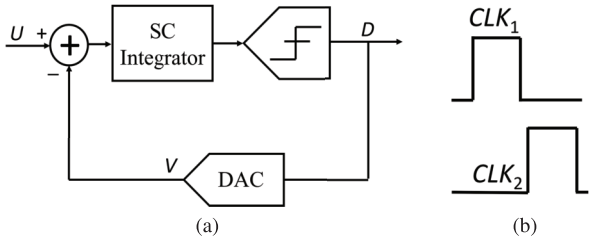


Fig. 21. The (a) block level schematic of a first-order delta-sigma modulator implemented on the FPAA and (b) timing profile of the clock signal applied to the modulator.

V_{INN} represent a differential input voltage signal, the voltages V_{RP1} and V_{RN1} represent a differential reference voltage signal, and the voltage V_{CM} represents the common-mode voltage signal. Each comparator and corresponding sampling network is implemented on a single CAB. Programmable TPs within the CAB are utilized to implement the comparator, while capacitors C_1 and C_2, which are shown in Fig. 2, serve as the sampling network.

The sampling phase of the 1.5-bit ADC is synchronized with the MDAC. Following the falling edge of CLK_{2P}, the sub-ADC generates the digital code that controls the analog output voltage from the DAC. Each pipeline stage is implemented within a dedicated column of the CAB matrix, which is shown in Fig. 1. The top row of CABs is allocated to the implementation of the DAC, while the middle and bottom rows are allocated to the comparators. To implement a three-stage pipeline ADC on the FPAA, all three columns of the CAB matrix are needed, where each stage of the ADC is mapped to a single column. The digital output from each stage is routed to a decoder, which generates the final binary output of the pipeline ADC. The analysis of the FFT spectrum of the configured pipeline ADC, sampled at a frequency of 15 MHz, is shown in Fig. 20. The FFT spectrum indicates an effective number of bits (ENOB) of 3.76. The power consumption of the configured pipeline ADC is 6.86 mW.

B. First-Order Delta-Sigma Modulator

Delta-sigma ADCs [31] provide high-resolution low-frequency analog signal quantization, which is a desired property. The block level representation of a first-order delta-sigma modulator (ΔΣ-MOD1) is shown in Fig. 21. The transfer function of a ΔΣ-MOD1 is given by

$$V(Z) = Z^{-1}U(Z) + (1 - Z^{-1})E(Z), \tag{3}$$

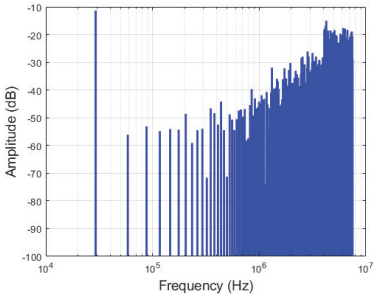


Fig. 22. Characterization of the FFT spectrum of the configured ΔΣ modulator. The frequency of the input signal is 29 kHz and the sampling frequency is 15 MHz, providing an oversampling ratio of 128.

where $V(Z)$ represents the output signal, $U(Z)$ represents the input signal, and $E(Z)$ represents the quantization error. The signal transfer function STF and the noise transfer function (NTF) are, therefore, given by

$$\mathrm{STF} = Z^{-1}, \quad \text{and} \tag{4}$$

$$\mathrm{NTF} = \left(1 - Z^{-1}\right), \tag{5}$$

respectively, where Z^{-1} describes a delayed version of the input signal Z in the frequency domain. The NTF exhibits a high-pass response, effectively shifting the noise spectrum to higher frequencies. Therefore, the first-order delta-sigma modulator allows for the use of decimation filters. In contrast, the signal transfer function STF indicates that the output signal is a time-delayed version of the original input signal. The discrete-time first-order delta-sigma modulator shown in Fig. 21 is implemented on the FPAA fabric. The delta-sigma modulator is comprised of a switch-capacitor integrator, a strongArm comparator, and an SR latch, as shown in Fig. 21. The detailed implementation of the switch-capacitor integrator and the strongArm comparator is described in Sections V-B and V-C, respectively. In addition, a 1-bit DAC, which provides voltage outputs of either $\mathrm{V_{DD}}$ or GND, is programmed in the same CAB as the integrator.

Given that each sub-block efficiently occupies one CAB, the delta-sigma modulator is integrated within the same column of the CAB matrix. Connections between the comparator and the integrator are established through the global SBs. The SR-latch is implemented on one block of the CLB array. The timing diagram of the clock of a ΔΣ-MOD1 is shown in Fig. 21. For the integrator, the voltage of the input is sampled by the sampling capacitor C_1 when $\mathrm{CLK_1}$ is high, while the reference voltage is integrated onto the output node when $\mathrm{CLK_2}$ is high. The comparator determines the feedback reference voltage applied to the integrator and outputs a resulting voltage (either 0 or 1.2 V) that is applied as input to a 1-bit DAC. Following the rising edge of $\mathrm{CLK_1}$, the differential output voltage from the integrator is applied as input to the comparator. If the output from the comparator is V_{DD}, a voltage of $-V_{\mathrm{REF}}$ is applied as input U and is passed to the switched-capacitor (SC) integrator during the subsequent clock cycle. If, however, the output from the comparator is zero, a voltage of $+V_{\mathrm{REF}}$ is applied as input U and is passed to the SC integrator during the subsequent clock cycle. The sampling frequency of the configured ΔΣ-MOD1 is 15 MHz. When subjected to an input signal with a frequency of 29 KHz and a decimation ratio of 128, as shown in Fig. 22, the configured first-order delta-sigma modulator provides a

TABLE I
COMPARISON OF PROPOSED FPAA CHARACTERISTICS WITH PRIOR FPAA TOPOLOGIES

FPAA	Technology	Area	Storage Element	CAB Component	Num. of CABs / CLBs	Architecture	Programmability	Synthesis Tool	Applications
[7]	350 nm	84 mm^2	FG SRAM	OTAs, Transistors, FG T-gate multiply, 8, 4-inputs BLEs	98 / 98	FG, Rapid reconfiguration 16-bit microprocessor, Manhattan	359,014 tunable analog parameters	VPR/VTR, Placement, Routing	Linear-phase filter [21], Hopf bifurcation circuit [22], Adaptive-Q BPF [23]
[4]	350 nm	9 mm^2	FG SPI	OTAs, Transistors, FG	32 / 0	FG crossbar	50,000 tunable analog parameters	VPR/VTR, Placement, Routing	Receiver, Speech processor
[15]	65 nm	3.46 mm^2	Byte-wise memory SPI	Programmable Transistors	600 / 0	FPGA-based island style	3,368 width control bits 32,040 routing switches	FPGA-based, Placement, Routing	Amplifier, Bandgap reference
[12]	65 nm	2.41 mm^2	ALU	Crossing detectors, DACs, SR latches	16 / 100	FPGA-based island style	12,626 configuration bits	VPR/VTR, Placement, Routing	PLL, ADC, DC-DC converter
[32]	180 nm	5.5 mm^2	SRAM	S-AC units, S-AC DACs	8 / 0	Tile-based module	8 hidden layers, 156 hidden neurons, 256 output neurons	Stand cell-based ASIC, Shape Mapping, Cell Mapping	Neural Array, CNN, DNN
[33]	130 nm	-	LUT, Register	OTAs, Capacitors	2 / 0	Square-lattice, Crossbar, Manhattan	12 Tunable Building Block, 15 programmable biquadratic structures	Graph-based, OTA synthesis	Low-pass filter, High-pass filter, Band-pass filter
[16]	65 nm	0.1 mm^2	Shift Register	Configurable Transistor Pairs	36 / 0	Fixed global interconnect	360 topology control bits 72 tunable transistor pairs	CAB selection, Placement, Routing	Biquad filter, Ring Oscillator, Frequency Divider
This Work	65 nm	4 mm^2	Shift register	TP array, Resistors, Capacitors	12 / 4	Two-level Manhattan	2592 width control bits, 1080 topology control bits 216 tunable transistor pairs	Placement, Routing, Transistor Sizing	Amplifier, Integrator, Pipeline ADC, Delta-sigma modulator

resolution of 6.7 bits. The power consumption of the configured delta-sigma modulator is 1.84 mW.

VII. COMPARISON WITH FPAA PRIOR ART

A comparative analysis among various FPAA topologies has been conducted, with key metrics listed in Table I. The structure of the CABs constrains the granularity of the circuits programmed onto the FPAA. In [15], each CAB functions as a programmable transistor, which allows for transistor-level configuration. However, the scale of the implemented circuits is restricted to a relatively modest scope, allowing for the programming of only five-transistor OTAs and lower order *RC* filters. The FG-based FPAAs [4], [7] provide the highest density of tunable parameters, where the FG devices are utilized as memory elements in the FPAA and are also used to set the bias voltage of implemented analog circuits including OTAs. A switch-less interconnect between CABs is implemented in [16], which limits the parasitic effects due to the routing switches. Each CAB of the FPAA fabric includes two tunable TPs, with a target of implementing bandpass filters, ring oscillators, and frequency dividers. For the digital blocks of the FPAA described in [32], the CAB includes a shape-based analog computing (S-AC) unit to implement neural networks such as CNNs and GNNs. In [33], a cell-based design is proposed to generate different types of biquadratic filters on the FPAA fabric. A time-domain FPAA [12] is developed to emulate analog signals utilizing a CAB that includes a DAC, a crossing detector, current sources, and programmable capacitors. In [3], an open-source toolset is proposed to enable analog-digital hardware-software co-design on the FPAA. A place-and-route algorithm, VPR/VTR [1], is utilized to map the target circuit onto the FPAA fabric. In this work, the proposed placement and routing algorithm treats the TP as a fundamental unit for placement, and the TPs are routed as differential signals. The programmable topologies of a TP include a current mirror, a regeneration loop, a current source pair, and a diode-connected pair. The algorithm balances the parasitic impedance of analog circuits operating in differential mode. For example, the strongArm comparator is sensitive to an unbalanced parasitic impedance, which introduces an offset voltage. However, the proposed algorithm only considers placing an analog circuit onto a single CAB, with the target circuit constrained to fit within a 6×3 TP array. A predefined bitstream is required to configure the CLB into a target digital circuit. To automate the mapping of large mixed-signal circuits onto the programmable fabric, future research includes dividing a large circuit into smaller blocks, with each block fitting within the TP array and utilizing open-source place-and-route algorithms including VPR/VTR, which enables the implementation of large digital programmable circuits.

The programmable NMOS and PMOS TPs are treated as the fundamental components utilized to implement widely used analog circuit topologies including current mirrors, regeneration loops, current sources, and diode-connected pairs. The architecture of the TP array, which is comprised of three rows of PMOS TPs (PPs) and three rows of NMOS TPs (NPs), connects the source terminals of the PPs to V_{DD} and the source terminals of NPs to GND through routing switches. In comparison to the FPAA architecture described in [15] and [18], where both the drain and source terminals are connected to TG switches, the proposed architecture reduces the number of routing switches in the dc current path from V_{DD} to GND and increases the available voltage headroom of the implemented analog circuits. The proposed CAB architecture includes programmable passive devices to facilitate differential signal processing and supports the implementation of large mixed-signal circuits including pipeline ADCs and delta-sigma modulators. The 3×4 CAB array that includes a CLB in each column enables integration of subcircuits through global switch-boxes. All sub-blocks are configured entirely by programming TPs, without requiring any integration of ASIC-based analog circuits.

VIII. DISCUSSION AND USE CASES

The proposed FPAA topology provides both fine-grain transistor-level reconfiguration and coarse-grain system-level reconfiguration. Potential use of the FPAA includes adaptive analog interfaces in wearable and IoT [34] devices, which support a variety of signal types including temperature, pressure, light intensity, and chemical concentration. A variety of analog front-end circuits including op amps, transconductance amplifiers, and integrators are needed to process such diverse signals. Designing a generic front-end circuit to accommodate all sensor types is a challenge. An FPAA is, therefore, an ideal choice to implement adaptive analog front-ends for such applications. The transistor-level programmability of the TP array supports a large variety of analog circuits. The FPAA fabric is configured into a specific front-end circuit when such a front-end interface is required.

The implementation of a pipeline-ADC and delta-sigma ADC demonstrates the versatility of the FPAA in implementing circuits for audio signal processing. The FPAA provides a versatile platform for the implementation of ADCs, while also proving robust for a diverse set of signal conditions, including dynamic environments, frequencies, and noise. For high-resolution audio systems, such as studio recording or audiophile-grade playback, an FPAA is configured as a delta-sigma ADC to achieve high resolution. For high-speed audio applications, such as real-time sound mixing or low-latency voice processing, the FPAA is configured as a pipeline ADC to support higher sampling frequencies. In addition, the reconfigurability of the FPAA fabric enables adaptive filtering and noise shaping directly after the ADC, which allows for real-time optimization based on the audio source or environmental conditions.

Compared to discrete components and analog hard-wired ASICs, the proposed FPAA architecture supports the configuration of a wide range of analog circuits, including op amps, comparators, integrators, and even larger mixed-signal systems such as pipeline ADCs and delta-sigma modulators. By providing a fabric that allows for the programming of multiple analog functions into a single FPAA, the size and complexity of an analog circuit is reduced as compared to an implementation that utilizes discrete components. Treating the TP as the fundamental programmable element allows for the implementation of a greater number of analog functions. The transistor-level programmability and tunable transistor widths enable reconfiguration of circuit parameters and topologies postfabrication, making the FPAA particularly advantageous for prototyping, iterative design, and adaptive applications. The proposed synthesis flow automatically maps a target analog circuit onto the FPAA fabric and includes a resizing process to account for the parasitic impedance of routing switches. The automated synthesis flow eliminates the need for manual design and characterization of layouts, a process typically required for discrete components and analog hard-wired ASICs.

IX. Conclusion

In this article, an FPAA architecture is introduced, offering both transistor-level and system-level reconfiguration. The programmable TP array allows tuning of transistor sizes and modifications to the connections between transistors. The structure of the TP array enables the implementation of various differential-mode analog circuits. The proposed synthesis and physical design flow automatically maps a given analog circuit onto the FPAA fabric. In addition, a novel placement algorithm is developed that determines both the vertical and horizontal positions of the transistors of an analog circuit. The proposed CAB architecture supports operations in both continuous-time and discrete-time. The smaller circuits implemented on the FPAA fabric include a folded-cascode op amp, a StrongArm comparator, a continuous-time integrator, and a switch-capacitor integrator. Larger AMS circuits implemented on the proposed FPAA include a four-stage pipeline ADC and a first-order delta-sigma modulator.

The FPAA, which includes a 3×4 CAB array and a CLB in each column of the array, occupies an area of 4 mm^2 in a 65-nm TSMC fabrication process. The implemented folded-cascode op amp provides a programmable dc gain of 28.3 dB to 34.8 dB with a tunable 3-dB bandwidth of 3.3 MHz to 5.5 MHz. The programmed StrongArm comparator provides a resolution of 5 mV when comparing two signals. The implemented continuous-time integrator effectively samples a 250 MHz square wave signal, resulting in a triangular wave signal of equal frequency. The programmed switch-capacitor integrator operates at a frequency of 15 MHz, providing a step size of 30 mV. The configured pipeline ADC provides an ENOBs of 3.7 for a sampling frequency of 15 MHz, while the implemented delta-sigma modulator provides an ENOBs of 6.8 for a sampling frequency of 15 MHz and an oversampling ratio of 128. Applications suited for the proposed FPAA include adaptive analog interfaces for wearable and IoT devices and audio signal processing.

References

[1] J. Hasler, "Large-scale field-programmable analog arrays," *Proc. IEEE*, vol. 108, no. 8, pp. 1283–1302, Aug. 2020.

[2] J. Hasler, "The rise of SoC FPAA devices," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2022, pp. 1–8.

[3] M. Collins, J. Hasler, and S. George, "An open-source tool set enabling analog-digital-software co-design," *J. Low Power Electron. Appl.*, vol. 6, no. 1, p. 3, Feb. 2016.

[4] A. Basu et al., "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[5] C. R. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, Sep. 2012.

[6] R. B. Wunderlich, F. Adil, and P. Hasler, "Floating gate-based field programmable mixed-signal array," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1496–1505, Aug. 2013.

[7] S. George et al., "A programmable and configurable mixed-mode FPAA SoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2253–2261, Jun. 2016.

[8] S. Shah and J. Hasler, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 28–37, Mar. 2018.

[9] J. Hasler and E. Black, "Physical computing: Unifying real number computation to enable energy efficient computing," *J. Low Power Electron. Appl.*, vol. 11, no. 2, pp. 1–21, Mar. 2021.

[10] 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," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2759–2768, Dec. 2008.

[11] D. De Dorigo and Y. Manoli, "An OTA-C signal processing FPAA with 305 MHz GBW and integrated frequency-independent filter tuning," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2016, pp. 61–64.

[12] Y. Choi, Y. Lee, S.-H. Baek, S.-J. Lee, and J. Kim, "CHIMERA: A field-programmable mixed-signal IC with time-domain configurable analog blocks," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 431–444, Feb. 2018.

[13] Z. Chen and I. Savidis, "Reconfigurable analog array for hardware security," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 1047–1051.

[14] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee, "Zero-crossing detector based reconfigurable analog system," *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2478–2487, Nov. 2011.

[15] N. Suda, J. Suh, N. Hakim, Y. Cao, and B. Bakkaloglu, "A 65 nm programmable ANalog device array (PANDA) for analog circuit emulation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 2, pp. 181–190, Feb. 2016.

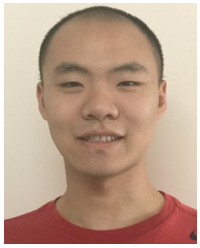
[16] Z. Chen and I. Savidis, "Reconfigurable array for analog applications," in *Proc. IEEE 39th Int. Conf. Comput. Design (ICCD)*, Oct. 2021, pp. 361–365.

[17] M. S. Diab and S. A. Mahmoud, "On the design of OTA-C based field programmable analog arrays for continuous time low frequency applications," *Microelectron. J.*, vol. 103, Sep. 2020, Art. no. 104870.

[18] J. Suh, N. Suda, C. Xu, N. Hakim, Y. Cao, and B. Bakkaloglu, "Programmable analog device array (PANDA): A methodology for transistor-level analog emulation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 6, pp. 1369–1380, Jun. 2013.

[19] B. Rumberg, D. W. Graham, and M. M. Navidi, "A regulated charge pump for tunneling floating-gate transistors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 3, pp. 516–527, Mar. 2017.

[20] A. Basu and P. E. Hasler, "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 953–962, Jun. 2011.
[21] J. Hasler and S. Shah, "An SoC FPAA based programmable, ladder-filter based, linear-phase analog filter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 592–602, Feb. 2021.
[22] J. Hasler, "A programmable on-chip Hopf bifurcation circuit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 12, pp. 4958–4968, Dec. 2022.
[23] J. Hasler, "A programmable adaptive-Q BPF circuit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 10, pp. 3888–3898, Oct. 2023.
[24] J. Hasler, P. R. Ayyappan, A. Ige, and P. Mathews, "A 130 nm CMOS programmable analog standard cell library," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 6, pp. 2497–2510, Jun. 2024.
[25] E. Strasnick, M. Agrawala, and S. Follmer, "Scanalog: Interactive design and debugging of analog circuits with programmable hardware," in *Proc. 30th Annu. ACM Symp. User Interface Softw. Technol.*, Oct. 2017, no. 10, pp. 321–330.
[26] J. Hasler, "The potential of SoC FPAAs for emerging ultra-low-power machine learning," *J. Low Power Electron. Appl.*, vol. 12, no. 2, p. 33, Jun. 2022.
[27] Z. Chen and I. Savidis, "Block configuration algorithms for a reconfigurable analog array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2024, pp. 1–5.
[28] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *J. ACM*, vol. 20, no. 3, pp. 385–388, Jul. 1973.
[29] S. Mallya and J. H. Nevin, "Design procedures for a fully differential folded-cascode CMOS operational amplifier," *IEEE J. Solid-State Circuits*, vol. 24, no. 6, pp. 1737–1740, Dec. 1989.
[30] Y. Chiu, P. R. Gray, and B. Nikolic, "A 14-b 12-MS/s CMOS pipeline ADC with over 100-dB SFDR," *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2139–2151, Dec. 2004.
[31] Z. Tan, C.-H. Chen, Y. Chae, and G. C. Temes, "Incremental delta-sigma ADCs: A tutorial review," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4161–4173, Dec. 2020.
[32] P. Kumar et al., "ARYABHAT: A digital-like field programmable analog computing array for edge AI," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 5, pp. 2252–2265, May 2024.
[33] B. Ray, D. Datta, M. Bhanja, and A. Banerjee, "Cell-based synthesis of multiple analog filter and oscillator topologies employing graph," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5152–5168, Dec. 2022.
[34] A. M. Rahmani, W. Szu-Han, K. Yu-Hsuan, and M. Haghparast, "The Internet of Things for applications in wearable technology," *IEEE Access*, vol. 10, pp. 123579–123594, 2022.



Ziyi Chen received the bachelor's degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2017, and the master's degree from Northeastern University, Boston, MA, USA, in 2019. He is currently working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA.

His research interests include field-programmable analog arrays (FPAA), side-channel attacks on the analog circuits, and analog obfuscation techniques using FPAA.

Ioannis Savidis (Senior Member, IEEE) received the B.S.E. degree in electrical and computer engineering and biomedical engineering from Duke University, Durham, NC, USA, in 2005, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2007 and 2013, respectively.

He joined the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA, in 2013, where he is currently an Associate Professor and directs the Integrated Circuits and Electronics (ICE) Design and Analysis Laboratory. His current research interests include analysis, modeling, and design methodologies for high-performance digital and mixed-signal integrated circuits, power management for SoC and microprocessor circuits, hardware security, including digital and analog obfuscation and Trojan detection, and electrical and thermal modeling and characterization, signal and power integrity, and power and clock delivery for heterogeneous 2-D and 3-D circuits.

Dr. Savidis is a member of the Association of Computing Machinery, the IEEE Circuits and Systems Society, the IEEE Communications Society, and the IEEE Electron Devices Society. He was a recipient of the 2018 National Science Foundation Early Faculty (CAREER) Award. He serves on the Organizing Committees of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST) and the IEEE International Conference on Computer Design (ICCD) and the Steering Committee of the ACM Great Lakes Symposium on VLSI (GLSVLSI). He also serves on the Editorial Boards for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, *Microelectronics* journal, and *ACM Transactions on Design Automation of Electronic Systems*.