Embedded Classifiers for Energy-Constrained IoT Network Security

Jennifer Hasler

Abstract We discuss the impact of physical computing techniques to classifying network security issues for ultra-low power networked IoT devices. Energy-constrained IoT systems, such as wearable devices, are already sensor rich and processing/computation constrained. The digital energy efficiency wall constrains the amount of signal processing possible at energy-constrained nodes. One rarely has any computational resources left to consider network security, leaving devices exposed. Fortunately many of these devices have infrequent wireless communication with very constrained command structures, but they still exhibit a system vulnerability, particularly when monitoring or controlling physical infrastructure. Physical computing approaches enable at least a factor of 1000 improvement in computational energy efficiency empowering a new generation of local computational structures for embedded IoT devices. These techniques offer computational capability to address network security concerns.

1 Sensor Nodes Empowered by SoC FPAA Devices

Analog Computing has grown up, fueled through the emergence of large-scale Field-Programmable Analog Array (FPAA) devices (e.g., SoC FPAA [11]), the generalization of FPGAs. Physical computing [12], which includes analog computing, enables both improved computational efficiency (speed and/or larger complexity) of $\times$1000 or more compared to digital solutions (as predicted by [27]) and potential improvements in area efficiency of x100. Physical computing is now programmable and configurable (e.g., [11]). The rise of programmable and configurable analog techniques (e.g., [11]), integrated with digital processing, enables a wide use of physical computing techniques, not limiting to a few analog IC design specialists [11, 12].

J. Hasler (✉)
Georgia Institute of Technology, Atlanta, GA, USA
e-mail: jennifer.hasler@ece.gatech.edu

Ç. K. Koç (ed.), *Cyber-Physical Systems Security*,
https://doi.org/10.1007/978-3-319-98935-8_14

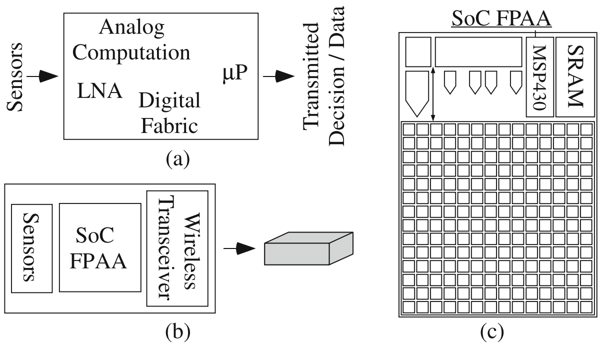


Fig. 1 Embedded configurable physical computation. (**a**) Physical computation in an embedded platform enabling a sea of analog and digital interacting and computation enable significant computing resources moving from sensors to decisions to be communicated. (**b**) Overview picture of the recently published SoC FPAA device [11]. (**c**) A wireless sensor node using this FPAA device, heavily utilizing context-aware techniques. The data from these experimentally measured structures will guide further scaling efforts (size, energy consumed). One application for this sensor network would be for ground-level monitoring of people, cars, trucks, machineries, or other elements through acoustic or MEMS vibration/accelerometer sensors. A second application for this sensor network would be for a body-level sensing network, monitoring the behavior of knees, heart, and other internal organs through a combination of vibrational and acoustic sensors

Figure 1 illustrates discussions on a wireless sensor node utilizing FPAA. FPAA devices allow the user to investigate many physical computing designs within a few weeks of time. The alternative for one design would require years of IC design by potentially multiple individuals. The sensor node could classify (e.g., [11]) and learn (e.g., [15]) from original sensor signals, performing all of the computation required for the computation and operating the entire system in its real-world application environment. Embedded learning approaches, implemented in a single FPAA device, illustrate the small area and ultra-low power capabilities of configurable physical computing. Section 2 discusses the context-aware opportunities in FPAA architectures.

Although the low-power physical computing could have huge impacts for network of autonomous sensor nodes, these FPAA-enabled nodes often require secure operation. Although FPAAs are a recent technology, widespread adoption of these devices eventually requires some level of security measures against malicious users. This discussion overviews low-power context-aware FPAA architectures (Sect. 2) and then addresses FPAAs as physical computing devices for low-power embedded applications (Sect. 3). The conversation moves to secure FPAA devices (Sect. 4), showing positive FPAA security attributes (Sect. 4.1) and addressing FPAA security issues (Sect. 4.2). FPAA devices can be used to investigate security of analog/mixed-signal capabilities (Sect. 5), as well as be part of the resulting secure computation, such as implementing unique functions (Sect. 5.3). The final section summarizes the discussions as well as addresses remaining issues for secure ultra-low power embedded FPAA devices.

2 Low-Power Context-Aware FPAA Architectures

Many portable and wearable devices are constrained by their energy efficiency. Figure 2 illustrates the energy impact for cloud computation, on-device digital computation, and FPAA-assisted computation. The digital communication typically dominates the overall energy consumption [14].

Cloud-based computing removes issues of real-time embedded (e.g., fixed-point arithmetic) to be done on some far away (and supposedly free) server using MATLAB-style coding. Computation done off-device is not seen and considered effectively endless; eventually that resulting energy and resulting infrastructure required still have significant impacts. The host system still must constantly transmit and receive data through its wireless communication system to perform these computations. The network connectivity must have a minimum quality at all times; otherwise performance noticeably drops. One often assumes the cloud is nearly free for a small number of users; as the product scales to the consumer market, these assumptions can break down. Although the local digital device computation (for a good wireless network) requires similar energy for cloud and on-device computation (at a 100MMAC(/s) level), physical computation, such as FPAA-empowered devices, enables factors of 1000x improvement in the overall power requirements.

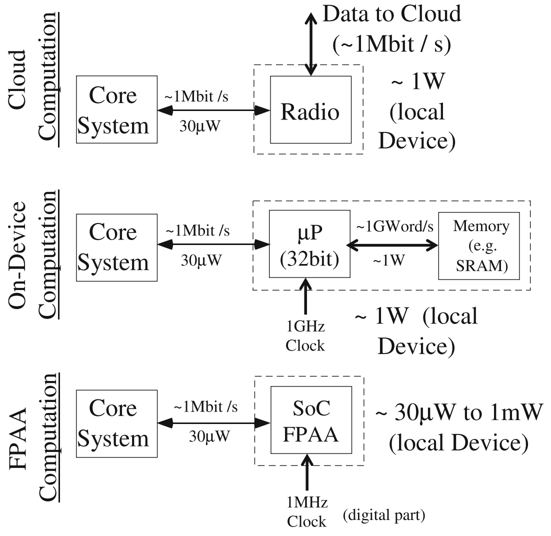


Fig. 2 Comparison of cloud computation, on-device computation, and FPAA computation. For cloud computation and for on-device computation, we only consider the energy required for communication. All devices might have an RF radio; we consider just the part required for this core computation. For FPAA computation we include the entire device. If cloud computation were considered free, then cloud and on-device computation would appear of similar complexity. FPAA computation dramatically decreases the resulting on-device computation

These approaches enable ultra-low power physical computing that enables small devices capable of computational-intensive ($>$10MMAC(/s)) context-aware processing. This approach requires low-power components, continuously operating or operating frequently, that can decide when to *wake up* the expensive hardware components. A node requiring $100\,\mu W$ average power could operate for several months on a single battery. These computing components are enabled by the x1000 energy improvement (and $\times$100 area improvement) from FPAA classification algorithms. The always-on computation in stage one requires being physical computation, both because of its computational power and its close proximity to sensor inputs.

The need for low-average power consumption requires that higher-power devices, like wireless transceivers and even embedded μP, must be shut down most of the time. These devices should be active only in those rare cases where they are needed, such as when messages need to be passed between nodes. Similarly high-power sensors and actuators (e.g., acoustic speaker) need to be shut down except when it is being used. Without using physical computation, the sensor node would be a simple, low-speed data acquisition node and likely cannot stay under 1mW given the power constraints of the embedded processor and wireless transceiver. The rest of the processing probably still needs to take place on some other digital system.

Physical computing in context-aware architectures enables potential energy-harvesting opportunities. Most energy-harvesting devices supply $\approx$10 μW of power per cm^2 except in unusual environments. Figure 3b shows the device lifetime (due to average power consumption) for a single coin cell battery (0.1–0.5 Ah). A 10 cm^2 energy-harvesting device could supply 100 μW of average power, a manageable area for an embedded sensor node.

3 FPAAs as Physical Computation Devices

FPAA devices are our vehicle for discussing ultra-low energy computing. FPAA devices allow the user to investigate many physical computing designs within a few weeks of time. The alternative for one design would require years of IC design by potentially multiple individuals. These FPAAs compare favorably against custom designs, and unlike FPGA designs, FPAA architectures are open to the academic community.

Floating-gate (FG) devices empower FPAA by providing a ubiquitous, small, dense, nonvolatile memory element [19] (Fig. 4). A single device can store a weight value, compute signal(s) with that weight value, and program or adapt that weight value, all in a single device available in standard CMOS [17, 18]. The circuit components involve FG-programmed transconductance amplifiers and transistors (and similar components) with current sources programmable over six orders of magnitude in current (and therefore time constant) [24]. Devices not used are programmed to require virtually zero power. FG devices enable programming

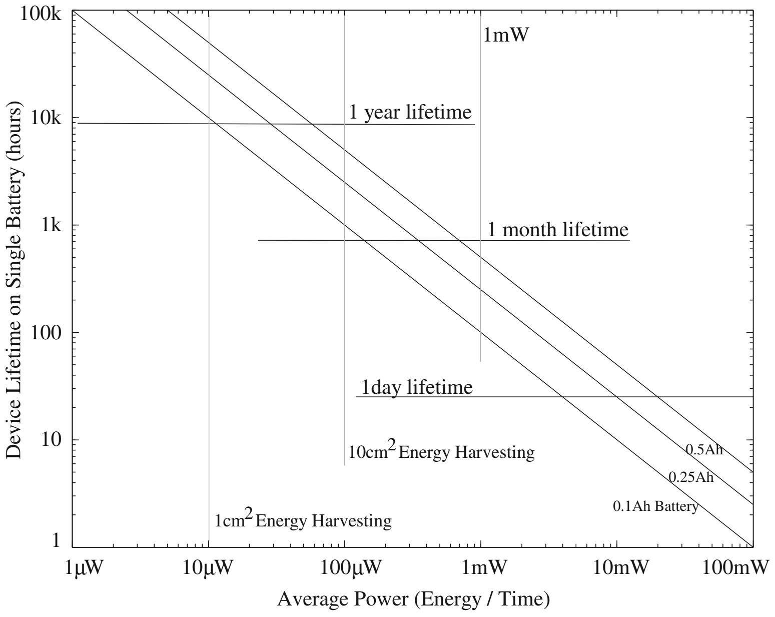


Fig. 3 Small sensor nodes require energy-efficient computation, computation enabled through physical computing approaches. Device lifetime available for wireless sensor nodes. The graph shows the opportunity for energy-harvesting systems at the size of 1 and 10 cm^2 form factors; most energy-harvesting systems output 10 μW of power per cm^2, with the exception of solar cells in direct sunlight in a desert

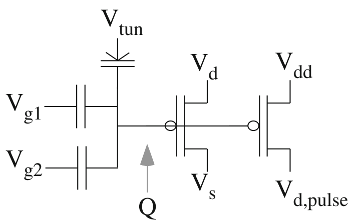


Fig. 4 Circuit diagram for basic FG device. The device can store a nonvolatile charge (Q), enables feedforward computation of functions involving Q, can program, and can adapt Q, all in a compact device structure. Multiple transistors sometimes ease the programming infrastructure for generic programming architectures

around device mismatch characteristics, enabling each device in a batch of ICs to perform similarly.

The SoC FPAA [11] ecosystem represents a device to system user-configurable system. An SoC FPAA implemented a command-word acoustic classifier utilizing hand-tuned weights demonstrating command-word recognition in less than 23 μW power utilizing standard digital interfaces (Fig. 5) [11]. Multiple analog signal processing functions are a factor of 1000× more efficient than digital processing,

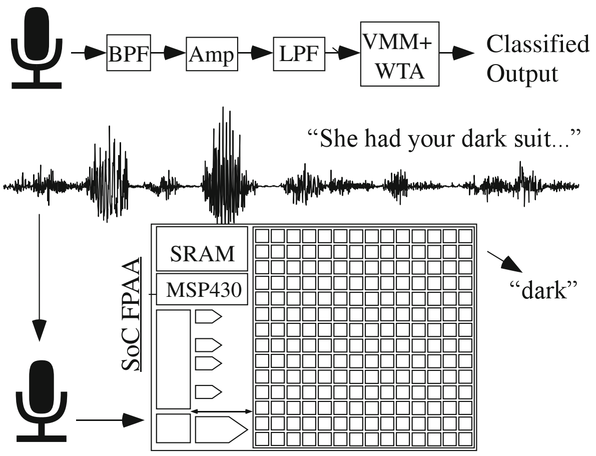


Fig. 5 High-level picture of low-power classification using an acoustic classifier for command-word classification. This approach enables computation from sensor to classified output in a single structure, handling all of the initial sensor processing and early-stage signal processing. The SoC FPAA device classified the word, such as *dark* from the TIMIT database phrases. This analog computation (<23 μW) is radically different than the class of expected analog operations

such as Vector-Matrix Multiplication (VMM), frequency decomposition, adaptive filtering, and classification (e.g., [11] and references within). Embedded classifiers have found initial success using this SoC FPAA device toward acoustic classification and learning (e.g., [11, 15]) in 10–30 μW average power consumption. The circuits compute from sensor to classified output in a single structure, handling all of the initial sensor processing and early-stage signal processing. This ecosystem will scale with newer ICs built to this standard, as expected by all future FPAA devices [20].

This new capability creates opportunities, but also creates design stress to address the resulting large co-design problem. The designer must choose the sensors as well as where to implement algorithms between the analog front end, analog signal processing blocks, classification (mixed-signal computation) which includes symbolic (e.g., digital) representations, digital computation blocks, and resulting μP computation. Moving heavy processing to analog computation tends to have less impact on signal line and substrate coupling to neighboring elements compared to digital systems, an issue often affecting the integration of analog components with mostly digital computing systems. Often the line between digital and analog computation is blurred, for example, for data converters or their more general concepts, analog classifier blocks that typically have digital outputs. The digital processor will be invaluable for bookkeeping functions, including interfacing, memory buffering, and related computations, as well as serial computations that are just better understood at the time of a particular design. Some heuristic concepts have been used previously, but far more research is required in building applications and the framework of these applications to enable co-design procedures in this space.

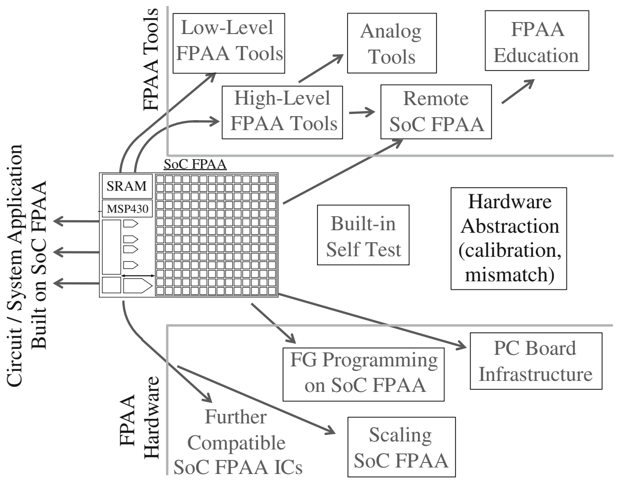


Fig. 6 SoC FPAA approach consists of key innovations in FPAA hardware, innovations and developments in FPAA tool structure, as well as innovations in the bridges between them. One typically focuses on what circuit and system applications can be built on the FPAA platform, but every solution is built up for a large number of components ideally abstracted away from the user

Analog computation [12] becomes relevant with the advent of FPAA devices, particularly the SoC FPAA devices [11]. Figure 6 shows a high-level view of the demonstrated infrastructure and tools for the SoC FPAA, from FG programming, device scaling, and PC board infrastructure through system-enabling technologies as calibration and built-in self-test methodologies and through high-level tools for design as well as education (e.g., [21]). This framework is essential for application-based system design using physical systems particularly given modern comfort with structured and automated digital design from code to working application. This framework utilizes abstractions in a mixed analog–digital framework [13], as well as development of high-level tools to enable non-device/analog circuit designers to effectively use these approaches [9] (Fig. 6). High-level design tools are implemented in Scilab/Xcos that enable automated compilation to working FPAA hardware [9]. These tools give the user the ability to create, model, and simulate analog and digital designs. Physical algorithms may show improvements beyond just energy efficiency for digital computing machines [12].

4 Embedded FPAA Security Concerns

The FPAA opportunities presented in the last section, particularly the ultra-low energy and small size characteristics, require consideration to make these embedded nodes secure. This section discusses multiple opportunities toward secure FPAA devices. Sections 4.1 and 4.1 discuss the positive characteristics and security issues, in turn, for the FPAA device family. Sections 5.1 and 5.2 discuss the aspects of using FPAA devices to develop training and procedures for deconstructing custom ICs, giving a sense of the security of a compromised FPAA device. Section 5.3 discusses using the FPAA infrastructure to build a unique function for security, potentially in securing the given FPAA device.

4.1 Positive FPAA Security Attributes

The FPAA structure has a number of good security aspects. The FPAA uses FG devices to store the device state without any SRAM loading vulnerability, particularly from an external IC. Once the FG values on the chip are programmed and loaded, the FPAA code is secure, unless one can scan out the states of the FG elements. FG programming an IC will have minimal changes over the lifetime (e.g., 10-year rating) of the part. The programming code is not the IC μP SRAM, but only used for programming, and then purged after programming. Analog values can be hard to measure without disturbing the values significantly, and digital computation can be encoded with analog computation and storage. Further, very low-power circuits are challenging to externally measure due to the low-circuit currents (e.g., pA and nA). These transistors do not have enough current or field to generate light to measure transistor behavior and become very hard to measure the external fields.

On the other hand, the FPAA structure is a platform for creating secure applications. The SoC FPAA structure is a generic structure, openly published, and built from general components. None of the particular components are unknown or confidential. IC layout says almost nothing about the programmed IC functions. The motivation to *steal* the knowledge of on-chip FPAA circuits is minimal. The infrastructure can measure the analog behavior at any given node in the FPAA. FPAAs allow for scanning every hardware node internally to the circuit (e.g., [11, 33]). If the core FG programming on the IC is verified, effectively part of the calibration procedure and measurement [25], then the entire IC can be verified. Secure analog and digital code can be programmed in a secure space.

The IC could have intelligence, using internal signals and voltages, to choose to erase its contents. If tampering is suspected, the operating device could pull up on the tunneling voltage line(s) in an attempt to erase the previous operating code. The device parallel erase occurs from a combination of electron tunneling and reverse tunneling. The result leaves little chance of recovering any previous code even with a short erase cycle. One is more likely to pick up device mismatch patterns rather than anything of the previous code.

4.2 Addressing FPAA Security Issues

FPAA devices are far from safe from a potential malicious agent, even with a number of good starting properties. For example, the current FPAA devices do not have encryption and related security on the input control of the device. If an actor could connect to the particular control connections, even if the IC pins are disconnected or disabled, they could get direct control of the device and programming infrastructure. Future FPAA devices will have encryption on the control structure, particularly as they move to a wider user community. The encrypted access can make use of a PUF from the particular FPAA, such as the approach shown in Sect. 5.3. Encryption is a straightforward solution used on secure FPGA devices. This section will consider the resulting issues for these devices.

Figure 7 illustrates possible security issues and types of attacks for an embedded system built with SoC FPAA device. The FPAA attacks could happen by physical tampering with an existing device, as well as electronic attacks through the communication port, such as a transceiver port. In a physical FPAA attack, the device is obtained while avoiding self-destruct sequence to be explicitly deconstructed. If the internal code can be obtained, likely at considerable expense, one could potentially reconstruct the FPAA function. Mismatch encoded functions would require additional computational and measurement structures. An alternate physical FPAA attack could use a compiled digital serial port to gain access to the digital control and resulting programming interface. When digital interfaces (e.g., SPI) are controlled by the processor, getting control of the processor is unlikely. A more likely situation is finding a way to stall the computation resulting from a physical attack on the clock structure. Many systems are far less secure due to physical tampering if the device has been obtained, and any self-destruct/erase mechanism was somehow avoided. A more likely situation is a nonphysical attack through the transceiver interface into the IC. These can include attacks to gain control of the FPAA device to reprogram the device or constantly attacking a device to drain the node battery power.

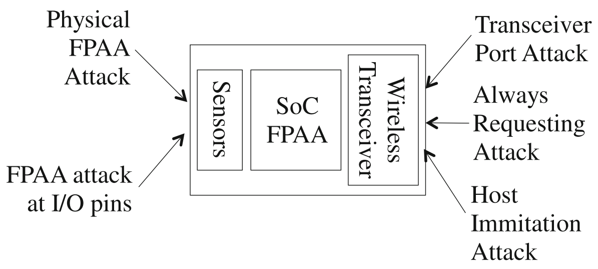


Fig. 7 Possible security issues for an embedded system built with SoC FPAA device. Some attacks could occur through the known communication path, such as through the wireless transceiver port, and other attaches could occur through direct physical access to the device

Table 1 Summary classification of IoT systems

Category	RAM	ROM
Class 2 (C_2)	50 kB	250 kB
Class 1 (C_1)	10 kB	100 kB
Class 0 (C_0)	<10 kB	<100 kB

Low-energy computation opens application opportunities at 10 mW, 1 mW, and lower-average power consumption, and yet the low power consumption constrains the system security capabilities. Embedded FPAA applications have limited digital memory because of the system cost. Network security is characterized in terms of classes of networked devices, summarized in Table 1 [6, 22, 26, 29, 31]. SoC FPAA is a C_0 device having only 32 kB total digital memory. Digital memory is expensive in terms of relative on-chip area, complexity, and energy dissipation. Many systems going forward might have less total digital memory, as well as many systems that will not rise to the C_1 memory level. FPAAs enable a whole opportunity of C_0 devices, devices many assume are impossible to secure over a network. Running a minimal OS and security code may exceed the rest of system energy budget.

So how do we have an ultra-low power secure IoT system? Part of the opportunity is coding systems outside of a minimal OS, consistent with the rest of the event-based FPAA μP code, as well as enabling tight secure stack and security aspects in MSP 430 assembly language. Digital FPAA event code is coded in assembly language and encapsulated in graphical code for easy user reuse.

5 FPAAs for Investigating IC Validation

When a user has a programmed FPAA device, it looks like any other custom IC that performs one or a set of functions. Further, the IC layout says nothing about the actual device performance. If the user knows it is an FPAA device and has sufficient knowledge of its programming functions, they might have additional information to figure out the function; otherwise, all they have is the device to characterize.

FPAA devices become good test platforms to investigate how individuals might deconstruct a particular IC. FPAA devices allow for many reprogrammed circuits, so the approach can be repeated many times. In the following subsections, we will discuss two such cases. First we will overview the inspiration of this study, the Black Box (BB) exam at Caltech (CNS 182). Second, we will discuss how this approach was modified, in an academic setting enabled by FPAA devices, to Training IC Deconstruction.

5.1 Black Box (BB) Exam: CNS 182

Academic groups are not usually interested in deconstructing known ICs, and when they are, the details are rarely discussed. One particular exception I personally experienced, both as a student (1993) and a teacher (1994–1996), was the Black Box (BB) exam at Caltech (CNS 182). This particular exercise was the final exam for the second quarter (Winter quarter) for CNS 182, Analog VLSI, and Neural Systems, between 1989 and 1996.

The exam consisted of a 2-h lab session followed by several days (4–5) to write up the results discovered during the lab session. The students in the class spent every week for two quarters measuring custom-built ICs, starting with transistors through small systems, using typical computer-controlled bench equipment. When the students arrived in the lab for the BB exam, a particular circuit consisting of 3–5 pins (besides power (V_{dd}) and ground (GND)) was operating correctly in one possible mode. Typically the circuit was a single transconductance amplifier (TA) or 2 TA circuit with a couple of transistors and known to be somewhat related to course topics over the first two quarters. This circuit was part of a 40-pin chip custom fabricated for the course; the students did not have access to any layout information. At least one element was a bias, set by a potentiometer. No FG devices were used. In the end, roughly half of the students would correctly guess the correct circuit with various levels of experimental justification.

5.2 Training IC Deconstruction Using FPAA BB Approach

The BB experience was recreated between 2011 and 2012 using currently available FPAA devices. The FPAA enables investigating deconstructing circuits, by providing a structured platform to instantiate a large number of circuits and systems. Each case would look from the IC pins to be some custom IC device and could be tested accordingly. The deconstruction capabilities can be quantified for different amounts of IC knowledge, such as routing information or netlists. These techniques could be used to verify a desired circuit implementation, as well as search for any additional component that was placed in the circuit. The FPAAs used for these experiments were designed between 2007 and 2010 (e.g., [4, 33]); the results should directly extend to using the SoC FPAA devices.

A group of graduate student IC designers were trained through a set of six BB events (Table 2) over a 9-month timeframe to eventually deconstruct a custom-fabricated IC. This BB approach arose from the constant interaction between courses and research. One person designed, compiled, and experimentally characterized the design completely without the knowledge of others. The groups had no idea of the functionality of the circuits before they arrived in the lab. Each person on the student team was previously familiar with measuring the FPAA devices. Between events students developed additional tools to assist in deconstructing the IC design.

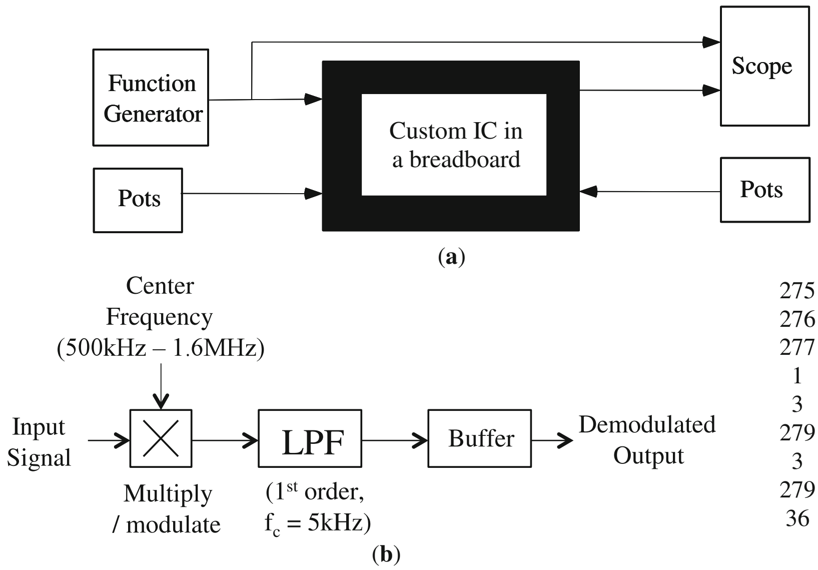
Table 2 Summary of FPAA black box experiments

	Components to find	Group info	Analysis techniques	Teams	Time
BB1	Analog Amps/muxes	Only IC	DC I/V, scopes	3 (2 people)	>8 h
BB2	Am Demod + hidden	Switch list	Switch list analysis + DC I/V, scopes	2 (3 people)	≈ 4 h
BB3	DAC: 5ibt R-22R + 3-bit V-mode = 8 bits	Netlist ≈100	Low-level netlists + DC I/V, scopes	2 (3 people)	7–8 h
BB4	Low-frequency transciever circuit	Netlist (spice)	Netlists, clustering, + DC I/V, scopes	2(4 people)	6–8 h
BB5	VCO controlled by 7-bit DAC	Netlist (spice)	Netlists, clustering, + DC I/V, scopes	2 (3 people)	5–6 h
BB6	Multiplexed 1 8bit DAC two in, two out	Netlist (spice)	Netlists, clustering, + DC I/V, scopes	2 (3 people)	4–5 h

Different events had different level of information (Table 2). The first case paralleled the Caltech experience to get a baseline performance, but with roughly double the number of chip pins and number of components, as well as the students involved did not prepare before this starting exercise. The groups did a number of I–V measurements at the chip pins to identify the resulting circuit. In the second case, the groups had a switch list (Fig. 8b), similar in format to the SoC FPAA approach [24]. The group made extensive use of the routing visualization tool, Routing Activity Tool (RAT), to uncover the resulting circuits. Whiteboard pictures prove this solution approach. Figure 8b shows the expected demodulation circuit which all groups found; the groups also found an unexpected extra oscillator that was explicitly added. In later cases the groups were given a form of netlist, compatible with the existing tools, for their analysis. All of the groups developed clustering algorithms to assist with grouping and identifying the resulting circuits. At each level, the speed to fully recognize and experimentally verify a particular circuit increased with the increasing circuit complexity.

The final goal was to extract and verify an entire custom IC developed by another group. A group of four Ph.D. students is involved in the BB experiences, and Dr. Hasler would spend three isolated days together to analyze this IC. Although the promised information varied throughout, in the end, the group was given (approximate) delayered information extracted from the IC, not including n-type or p-type selections. After 3 days and 2 additional days to write the report, the group found all four interleaved DACs, although only one was populated fully. The group discovered an error on the VCO due to a misplaced GND line.

This process showed FPAAs could be used to train individuals to deconstruct the circuitry on a particular device, as well as important insights to secure a particular FPAA device. Nonvolatile analog FG storage makes discovering the internal code of a programmed device extremely difficult without huge expenses. The approach showed some unique aspects of using physical computation related to security; the



275	35	1.8
276	24	1.8
277	25	1.8
1	51	1.8
3	45	1.8
279	62	1.8
3	88	1.8
279	81	1.8
36	8	1.8

(c)

Fig. 8 Illustration of the Black Box exam setup. (**a**) A student would arrive into the lab with a working device demonstrating some characteristic of the circuit. The question is to find the entire circuit, a circuit inside an integrated circuit with a few, e.g., I/O pins, in a finite amount of time (e.g., 2 h). This experience has parallels to security issues when deconstructing an unknown analog or mixed-mode circuit. (**b**) A low-frequency signal demodulator is an example system (BB2) to deconstruct from the available data. Each of these components was built using available CAB components and routed into the FPAA infrastructure. Typical electrical engineers might predict such an architecture when faced with multiple components. If an additional component is sitting in this circuit, it might create confusion or might just be overlooked. (**c**) In BB2, the groups had the switch list programmed into the FPAA device. The switch list communicates the physical routing on the FPAA IC. The first two columns are position in x and y direction on chip. The third column is log-encoded value for current level; 1.8 is a value to program as a switch

wider opportunities in physical computing [12] show these items are just scratching the surface of what is possible.

5.3 FPAAs for Unique Functions

Unique functions in FPAA IC devices are rich platforms to construct unique functions, particularly for security. The FPAA device allows for the selection of many devices, devices that have mismatch specific to a particular IC and mismatch that can be selected and compiled into a particular circuit. The mismatch between pFETs for a FG device enables almost 1M mismatched components.

Unique functions and PUFs have been implemented in FPGAs (digital) [28, 37] and analog circuits [8, 32]. For example, [37] uses delay variability in the FPGA

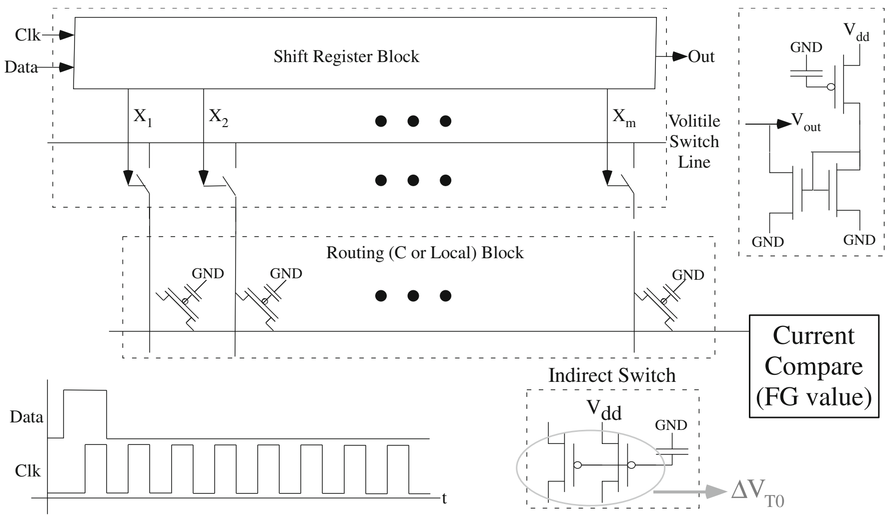


Fig. 9 Example of generating unique functions for secure codes implemented in an SoC FPAA. Threshold voltage (ΔV_{T0}) mismatch at a chosen location in nearly one million FG devices gives the resulting code

to create a specific code directly affected by the component variability. All of these functions are based on the mismatch of the resulting device, whether custom fabricated or compiled in the structure [36]. This FPAA approach is similar to the FPGA approach for making unique functions and PUF, in that a function is compiled on the device and utilized to create a unique output code for a particular input stimulus code.

Figure 9 shows an example of FPAA circuit for generating a unique function [16]. This approach utilizes the mismatch available in the FPAA circuit, mismatch we typically remove from the device. The structure yields a code for encryption of data, enabled by programming the desired code by the user. One use for the input code (stimulation) is the address of the FG elements to measure. The resulting outputs, scanned through shift registers available throughout the IC, would be thresholded to yield a digital code (Fig. 10). The FG elements would be programmed to bias the resulting code as desired, modulating the mismatch pattern. Typically one would program all elements to the same current to bring out the mismatch pattern (e.g., [11]). The programmed values would be retained for the operation of the FPAA IC, showing μV shift over a typical 10-year lifetime. The function could be compiled right into the rest of the circuitry, where implementation and routing of other circuits would obfuscate the resulting devices. This technique allows for an evolution of the codes through secure FG updates. If a code was suspected to be discovered, one could easily just move the sensing circuitry to an open circuit area. This unique function circuit may not have to be on the chip, but can be compiled onto a particular IC when needed [16]. If the IC is erased, knowledge of the PUF is erased except in the secure space originally used.

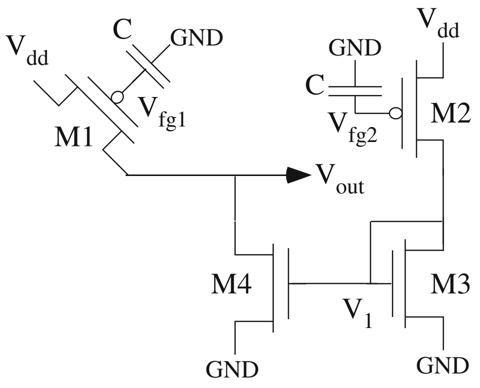


Fig. 10 Effective circuit diagram for the PVT analysis of the unique function circuit. The volatile switch line set to Vdd for this circuit

6 Summary and Next Directions

Physical computing opens great opportunities in energy-constrained IoT environments while creating significant security challenges for these IoT devices. FPAA devices enable the large-scale deployment of physical computation, and yet, these FPAA-enabled nodes often require secure operation against malicious users. Low-power context-aware FPAA architectures enable a number of autonomous sensor nodes. FPAA devices have a number of positive security attributes and security issues. FPAA devices can be used to investigate security and be part of the resulting secure computation.

We want to summarize current issues for building and deploying secure ultra-low power embedded FPAA devices. These directions include:

- Encrypt the control (and therefore programming) data stream, likely using a PUF circuit for the encryption code as part of the FPAA IC.
- Develop ultra-small security framework in dedicated assembly code + mixed-signal classification that integrates with event-based μP operation.

Network traffic attacks on FPAA-based systems are likely to be a point of vulnerability, requiring building tables and metrics of proper and improper network activity and classifying the resulting responses [1–3, 5, 7, 10, 23, 30, 34, 35]. These functions must be done in as low computational energy as possible. The functions require a minimal digital energy in parsing and creating these tables. Classification energy would be minimized using learning classifiers compiled on the FPAA infrastructure [15].

Security for ultra-low power embedded computing platforms based on FPAA devices is possible and is a space rich in potential research opportunities. The need for secure ultra-low power embedded computing platforms will likely only grow in the near future.

Acknowledgements This chapter resulted from a conference on IoT and embedded network device security in Paris in July 2017. My first exposure to security of embedded network devices started in Paris, not too far from the conference in 2017, when I (J. Hasler) visited Francois Bayen and his family in Paris in 1991 as a young graduate student. We had a number of technical discussions on embedded integrated circuit (IC) security related to a number of his European projects. These discussions gave me the background and interest to dig deeper in these areas when the time came; I am thankful for everything I learned from him during that visit and later visits. Twenty-six years later these discussions have come full circle. Writing this chapter on security of embedded devices, particularly FPAA devices, has brought me back to the knowledge source. Their family has remained close friends with my family to this day.

References

1. V. Asharani, B.N. Veerappa, M. Rafi, Security evaluation of pattern classifiers in adversarial environments. Int. J. Comput. Sci. Mobile Comput. **4**(4), 768–774 (2015)
2. T. Bakhshi, B. Ghita, On internet traffic classification: a two-phased machine learning approach. J Comput. Netw. Commun. 1–21 (2016)
3. R. Bar-Yanai, M. Langberg, D. Peleg, L. Roditty, Realtime classification for encrypted traffic, in *SEA* (2010), pp. 373–385
4. A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, P. Hasler, A floating-gate-based field-programmable analog array. IEEE J. Solid-State Circuits **45**(9), 1781–1794 (2010)
5. B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack. IEEE Trans. Knowl. Data Eng. **26**(4), 984–996 (2014)
6. C. Bormann, M. Ersue, A. Ericsson, Terminology for constrained-node networks, in *RFC 7228*. Internet Engineering Task Force (IETF) (2014)
7. A.A. Cardenas, J.S. Baras, Evaluation of classifiers: practical considerations for security applications (2006)
8. R. Chakraborty, C. Lamech D. Acharyya, J. Plusquellic, A transmission gate physical unclonable function and on-chip voltage- to-digital conversion technique, in *IEEE DAC* (2013), pp. 59:1–59:10
9. M. Collins, J. Hasler, S. George, An open-source toolset enabling analog–digital software codesign. J. Low Power Electron. Appl. **6**(1), 3 (2016)
10. A. Dainotti, A. Pescape, K.C. Claffy, Issues and future directions in traffic classification. IEEE Netw. **26**(1), 35–40 (2012)
11. S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R, Wunderlich, S. Nease, S. Ramakrishnan, A programmable and configurable mixed-mode FPAA SoC. IEEE Trans. Very Large Scale Integr. **24**(6), 2253–2261 (2016)
12. J. Hasler, Opportunities in physical computing driven by analog realization, in *IEEE ICRC, San Diego* (2016)
13. J. Hasler, Analog Abstraction, Computation, and Numerical Analysis. Accepted to ISCAS 2018, Florance, May 2018
14. J. Hasler, B. Marr, Finding a roadmap to achieve large neuromorphic hardware systems. Front. Neuromorphic Eng. (2013), 1–29. https://doi.org/10.3389/fnins.2013.00118
15. J. Hasler, S. Shah, SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier, in *IEEE ECAS* (2018)
16. J. Hasler, S. Shah, Security implications for ultra-low power configurable SoC FPAA embedded systems. J. Low-Power Electron. Appl. **8**(2), 1–18 (2018)
17. P. Hasler, C. Diorio, B.A. Minch, C.A. Mead, Single transistor learning synapses, in *Advances in Neural Information Processing Systems*, ed. by G. Tesauro, D.S. Touretzky, T.K. Leen, vol. 7 (MIT Press, Cambridge, 1994), pp. 817–824

18. P. Hasler, C. Diorio, B. Minch, C. Mead, Single transistor learning synapse with long term storage. IEEE Int. Symp. Circuits Syst. **3**, 1660–1663 (1995)
19. P. Hasler, B. Minch, C. Diorio, Adaptive circuits using pFET floating-gate devices, in *Advanced Research in VLSI* (1999), pp. 215–229
20. J. Hasler, S. Kim, F. Adil, Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems. J. Low Power Electron. Appl. **6**(3), 13 (2016)
21. J. Hasler, A. Natarajan, S. Shah, S. Kim, SoC FPAA immersed junior level circuits course, in *MSE* (2017)
22. T. Heer, O. Garcia-Morchon, R. Hummen, S.L. Keoh, S.S. Kumar, K. Wehrle, Security challenges in the IP-based internet of things? Wirel. Pers. Commun. **61**(3), 527–542 (2011)
23. Z. Khorshidpour, S. Hashemi, A. Hamzeh, Learning a secure classifier against evasion attack, in *IEEE Data Mining Workshops* (2016), pp. 295–302
24. S. Kim, J. Hasler, S. George, Integrated floating-gate programming environment for system-level ICs. EEE Trans. Very Large Scale Integr. **24**(6), 2244–2252 (2016)
25. S. Kim, S. Shah, J. Hasler, Calibration of floating-gate SoC FPAA system. IEEE Trans. Very Large Scale Integr. (2017)
26. J. King, A.I. Awad, A distributed security mechanism for resource-constrained IoT devices. Informatica **40**, 133–143 (2016)
27. C. Mead, Neuromorphic electronic systems. Proc. IEEE **78**, 1629–1636 (1990)
28. N. Muthumeenakshi, S. Sharma, M. Farjanaaameera, R. Rajaprabha, LSI design of low cost (PUF) physical unclonable function using FPGA and highly secured clock network. IOSR J. VLSI Signal Process. **4**(1), 16–21 (2014)
29. G. Oikonomou, I. Phillips, Experiences from porting the Contiki operating system to a popular hardware platform, in *IEEE DCOSS Workshop, Barcelona* (2011)
30. T. Omrani, A. Dallali, B.C. Rhaimi, J. Fattahi, Fusion of ANN and SVM classifiers for network attack detection, in *Sciences and Techniques of Automatic Control* (2018)
31. R. Roman, P. Najera, J. Lopez, Securing the Internet of Things. IEEE Comput. **44**(9), 51–58 (2011)
32. T. Saha, V. Sehwag, TV-PUF : a fast lightweight analog physical unclonable function, in *IEEE Nanoelectric and Information Systems* (2016)
33. C. Schlottmann, S. Shapero, S. Nease, P. Hasler, A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing. IEEE J. Solid-State Circuits **47**(9), 2174–2184 (2012)
34. P. Velan, M. Cermak, P. Celeda, M. Drasar, A survey of methods for encrypted traffic classification and analysis. Int. J. Netw. Manage. **25**(5), 355–374 (2014)
35. Z. Wang, The applications of deep learning on traffic identification
36. Z. Wang, Y. Chen, A. Patil, J. Jayabalan, X. Zhang, C.H. Chang, A. Basu, Current mirror array: a novel circuit topology for combining physical unclonable function and machine learning, in *IEEE TCAS I* (2017)
37. T. Xu, M. Potkonjak, Robust and flexible FPGA-based digital PUF, in *FPL* (2014)