Fabrication Security and Trust of Domain-Specific ASIC Processors

Michael Vai, Karen Gettings, and Theodore Lyszczarz
MIT Lincoln Laboratory
Lexington, MA, USA
{mvai, karen.gettings, tedl}@ll.mit.edu

Abstract
This paper describes our experience developing techniques to protect embedded intellectual property (IP) while an ASIC is being fabricated in an untrusted foundry. We created a customizable, high performance, domain-specific ASIC processor architecture, which we showed to be effective in protecting IP and mitigating the expense and inflexibility associated with using ASIC technology. Using an ASIC Fast Fourier Transform (FFT) accelerator as a test case, we have investigated various obfuscation options and their practicality in ensuring the trust and security of the processor when it is fabricated. The result is a processor architecture that incorporates split fabrication, configurable switch arrays and fabrics, programmable controllers, and configurable functional kernels. We have introduced a quantitative metric to gauge the effectiveness of application obfuscation for a domain-specific processor during fabrication.

Keywords—Split Fabrication; Configurability; Programmability; Untrusted Fabrication; ASIC Processor; IP Protection; Obfuscation.

I. Introduction

Embedded systems can be built with General Programmable Processors (GPPs), Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs), or a combination of them. As each of these technologies provides a unique level of capability and flexibility, choosing the right one to meet the performance requirement without breaking SWaP (size, weight, and power) budget is critical in the development of embedded systems.

Figure 1 represents a summary result from our long-term survey of defense-related embedded applications, which shows the potential volume efficiency (e.g., computations per liter) and power efficiency (e.g., computations per watt) of three technologies. As described by Moore's Law, these values change as semiconductor technology advances, thus they are normalized in Figure 1 using the corresponding FPGA capability as a baseline. A variety of representative defense applications are placed on the chart according to their requirements.

Besides performance, embedded system developers must also consider design and manufacturing complexity and cost, availability and usability of fabrication technology, and the protection of critical intellectual property (IP) embedded in the design. For example, an ASIC processor potentially has a 10-1,000X performance advantage over its FPGA and GPP counterparts, but it is expensive and inflexible. Most of all, ASIC developers must mitigate the risks incurred in untrusted fabrication, which include IP theft, over-production for gray/black market sales, and unauthorized alteration. This paper describes the risk assessment and mitigation in our creation of a domain-specific ASIC processor.

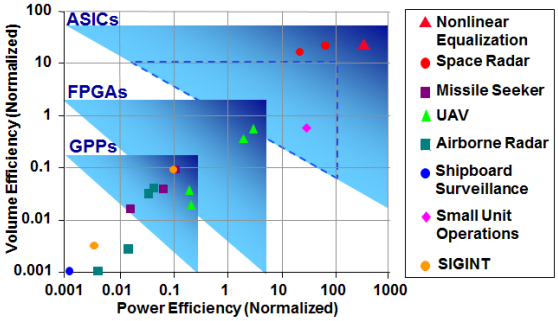

Figure 1: Embedded processing technology capabilities with the efficiency values normalized to account for Moore's Law (UAV: unmanned aerial vehicle; SIGINT: signals intelligence)

The rest of the paper is organized as follows. Section II explains our project of creating a domain-specific ASIC processor. Section III discusses a threat model for fabricating these processors in an untrusted foundry and an investigation of applicable obfuscation techniques. We then describe two domain-specific processor implementation variants for different usage models and explain the reasoning for our choices of obfuscation techniques. We introduce in Section IV a quantitative obfuscation metric that we have used to guide and assess our design. We then analyze, for each of the domain-specific processor variants, its performance benefits and its effectiveness in obfuscating critical IPs. We finish by discussing potential mitigations for identified residual vulnerabilities. Section V concludes this paper by summarizing our lessons learned from this project and suggests a few research directions.

II. Domain-Specific ASIC Processors

As Figure 1 has shown, the ASIC is indispensable in certain application areas, such as advanced signal processing embedded in small platforms. Besides the potential of higher performance than FPGA and GPP, the possibility of fabricating special functions (e.g., analog circuits, non-volatile memories, etc.) and digital functions on the same chip is very attractive in our embedded applications.

This work was sponsored by the Assistant Secretary of Defense for Research & Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government. Distribution Statement A. Approved for public release: Distribution unlimited.

Nevertheless, designers shy away from ASIC processors because they are expensive, inflexible, and take a long time to fabricate. Unlike commercial products (e.g., smart phones), our defense-related applications do not have the production quantity to amortize ASIC design costs. Furthermore, with the increasingly high cost of trusted foundries, more and more advanced semiconductor processes will only be available at untrusted foundries. Therefore, the technology protection concerns for ASICs will continue to grow with the lack of trusted manufacturing and the proliferation of inexpensive reverse engineering capabilities.

We thus have pursued a design methodology that produces domain-specific processors with close to ASIC performance, and we have addressed the other issues of using ASICs, such as their long fabrication time, high costs, and the need for IP security. Our approach is applicable to a broad class of defense-related applications (e.g., synthetic aperture radars) that depend on a small set of signal processing kernels, as illustrated in Figure 2. Note that these kernels must be configurable to support the parameters required by a wide range of target applications.

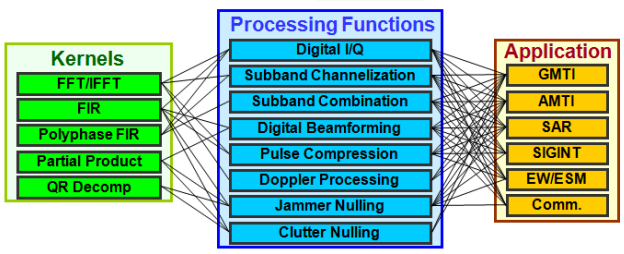

Figure 2: Defense applications are built with a few processing functions and kernels (GMTI: ground moving target indication; AMTI: airborne moving target indication; SAR: synthetic aperture radar; SIGINT: signals intelligence; EW/ESM: electronic warfare/electronic support measures; Comm.: communications)

The domain-specific ASIC processors under development are System-on-Chips (SoCs) consisting of configurable kernels interconnected with configurable fabrics and controlled by programmable processors. Other components such as analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and memories can also be included as required by a target application domain. Figure 3 illustrates an example of such a domain-specific processor, which can be customized for individual applications.

The configurable kernels (FIR, FFT, PPF, PP, and QR) deliver the ASIC performance needed for the applications. The algorithm (i.e., the so-called secret sauce) of a specific application will be defined by kernel configurations and connectivity as well as by the sequence of operations and data flow. In our design, the connectivity is provided by the configurable interconnection fabric and the operations are controlled by the programmable processor(s). This hardware-for-performance and software-for-flexibility model has been used successfully on a number of applications. For example, in the synthesizable cryptographic and key management processor described in [1], all cryptographic kernels have been implemented in hardware for high performance while the cryptographic protocols were implemented by software in a microcontroller for flexibility.

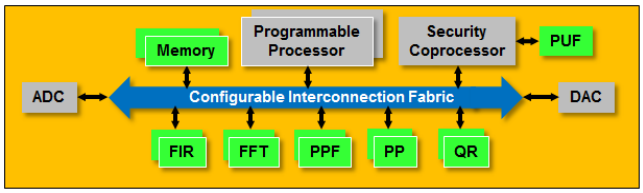

Figure 3: Example domain-specific customizable processor architecture (ADC: analog-to-digital converter; DAC: digital-to-analog converter; FIR: finite impulse response filter; FFT: fast Fourier transform; PPF: polyphase filter; PP: partial product; QR: QR matrix decomposition; PUF: physical unclonable function)

A security coprocessor and a physical unclonable function (PUF) are included in the architecture for *in-field* protection. The details of their operation (e.g., providing secure boot, root-of-trust, and encryption of configuration) were described in [2]. In this paper, we focus on the following question. How do we architect a domain-specific processor to protect application details when it is fabricated in an untrusted manufacturing facility?

III. Implementation for Untrusted Fabrication

Most circuit obfuscation research reported in literature takes a bottom-up approach, in which the obfuscation techniques are studied and/or assessed for general-purpose uses, without considering their mission and application specific roles, e.g., [3][4]. In addition, many of the studies are performed using low complexity benchmarks (e.g., ISCAS85 and ISCAS89) [5]. These studies have apparently been contributing to the development of a knowledge foundation. However, they offer very little insight into the practicality of various obfuscation techniques in a mission or application where decision makers would need to know: which obfuscation techniques can be used, to what extent, what is the risk, and what are the mitigations.

We have taken a "top-down" study approach and use mission and application requirements to drive the system design. For a domain of applications, how do we select obfuscation techniques to mitigate the risk of reverse engineering and repurposing during fabrication? Also, what is the impact on performance, area, power, and cost? The ultimate objective is to develop this knowledge and provide designers with a well-thought-out, multi-dimensional design space to ensure fabrication security and trust.

In order to provide a concrete example to drive our analysis, we decided to implement one of the computational kernels shown in Figure 3. Specifically, we chose to focus on a 1 million (1M) point FFT ASIC kernel. Due to its wide use in our signal processing applications, an FFT, in particular a large FFT, is a reasonable representative kernel in our domain-specific processors. We decided to implement the so-called sparse FFT (SFFT) architecture, which leverages the fact that frequency sparse signals can be

represented by a few non-zero Fourier coefficients [6]. The functional details of this FFT architecture were previously reported in [7]. This paper discusses its security and trust during fabrication.

The FFT kernel was first prototyped with an FPGA to establish a baseline for performance evaluation. In addition, we designed and fabricated an ASIC implementation in a 180-nm CMOS technology; see Figure 4. Compared with the FPGA version, the ASIC kernel achieved a 10X improvement in throughput at merely 10% of the power consumption, thereby achieving a 100X power efficiency gain. This supports our decision to develop domain-specific ASIC processors. In addition to 1M point sparse FFT, the kernel can be configured to perform 4 regular FFT operations, i.e., 256 point, 1K point, 4K point, and 16K point. Up to two smaller (i.e., 256 point and 1K point) FFTs can be configured to operate concurrently in the architecture.

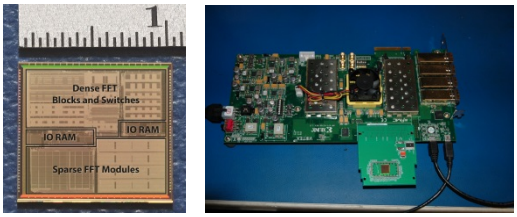

Figure 4: 1M point FFT ASIC kernel (as a standalone 1 cm^2 chip) and its testbed

A. Threat Model

The threat model reflects the focus of our study, which is fabrication security and trust for defense-related processors. In our application domains, one of the top concerns is the possibility of reverse engineering of system capability from the embedded processor. For example, the resolution of a sensor system may be inferred from the size of its FFT operations, which is another reason we have studied the obfuscation of an FFT. An equally important objective is the protection of advanced algorithms (i.e., the secret sauce) embedded in the IC, which can be expressed as a series of computational steps with signal processing kernels.

In contrast, the designs of kernel accelerators, such as FFT, FIR, QR, etc., are often well developed and available as textbook examples, in many cases through either open source archives or commercial IPs. We thus focused on obfuscating their application specific parameters (e.g., FFT sizes, FIR filter coefficients, etc.) during fabrication to avoid revealing application capabilities. The system level kernel connectivity and orchestration for operation (e.g., a process for adaptive beamforming) must also be protected from the access by an untrusted manufacturer.

Unauthorized modifications such as the insertion of Trojan circuits that affect operations and/or reliability are also a major concern. Overproduction for gray/black market sales is also a threat.

B. Mitigation

The purpose of design obfuscation is to address the listed security concerns in light of the information resources supplied to the foundry. Our study has focused on the protection of system capabilities and secret sauce. Obfuscation also helps with mitigating the threats of unauthorized modification and overproduction, which will be discussed in Section IV as residual vulnerabilities.

The obfuscation technologies considered in our study are split fabrication, logic obfuscation, configurable connectivity, configurable functions, and programmable operations.

C. Split Fabrication

Our original plan considered only split fabrication [8] in which an untrusted foundry is tasked with the so-called front end process of fabricating the transistors and the first level of metal. A second, trusted foundry, completes the back end fabrication of the remaining layers of metal interconnect.

We planned to use only upper metal layers in the back end process to customize a domain processor for specific applications. This approach would provide application flexibility, quick-turn-around customization, and obfuscation of functionality. However, our analysis and simulation quickly showed that this scheme of solely depending on split fabrication incurs serious design complexity and performance penalties. The widths and pitches of upper metal layer interconnections are significantly more restrictive than their lower layer counterparts. For example, in 45 nm CMOS, the 9th and 10th metal layers have a pitch of 1,600 nm, while the 2nd and 3rd metal layers have a pitch of 140 nm. In addition to coarser interconnect granularity, the wider upper layer metal interconnects have higher capacitance, and this causes increased delays and requires the kernels to have larger drivers. In addition, the efficacy of the obfuscation will depend on where the fabrication process is split. Limiting the untrusted foundry to fabricating the transistors and first layer of metal results in a strong obfuscation, but it requires the trusted foundry to fabricate several layers of dense metal and this will be impractical unless the trusted foundry supports advanced fabrication. On the other hand, fabricating all the dense layers at the untrusted foundry reveals too much of the design.

Other issues are the complexity of coordinating the processes, increased costs, lack of availability of split fabrication multiproject runs, and the potential speed and area penalties caused by routing critical signals in the upper level metals for the purpose of obfuscation. We did not undertake a quantitative evaluation of the return-on-investment on split fabrication, but instead relied on information available in literature, e.g., [9]-[13]. Based on these considerations, we developed the modified split fabrication architecture described below.

Instead of using multiple top-level metal layers to directly interconnect the kernels, we created (in the front end process) dedicated routing lanes embedded between kernels using the dense local and intermediate metal layers. This concept is shown in Figure 5. Pass-transistor switches controlled by upper level metal layer connections (to be made in the back end process) are used to physically complete the routing connections. With this structure, the back end process will

configure and connect the kernels indirectly by selectively turning on switches. All the high-speed signals are confined to the lower metal layers to maximize performance. The area overhead of these routing lanes scales with the footprints of kernels they are connecting, but in general the routing will be negligible since the kernels are typically quite large. For the FFT design shown in Figure 4, the switches represented only 1% of the total chip area. This switch-based customization scheme is easier to design and has significant area and delay advantages over the original upper metal configuration scheme.

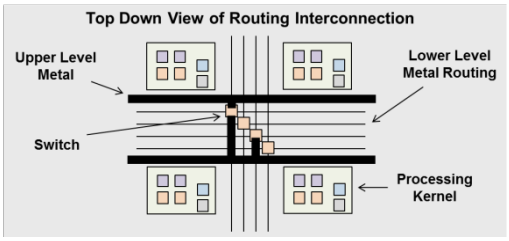

Figure 5: Configurable routing structure that overcomes the upper level metal routing challenges in split fabrication

An alternative split fabrication approach is to use 2.5D and 3D integration techniques that allow the integration of two or more chips either by direct wafer bonding or advanced packaging, see [14][15]. 2.5D typically refers to the use of silicon interposers between a substrate and components, where the interposer has through-silicon vias (TSVs) connecting the metallization layers on its upper and lower surfaces, while 3D IC connect components directly through TSVs, without the use of interposers [15]. A range of vertical integration approaches can be implemented by bonding together full wafers, stacking individual circuit die, or employing hybrid die-to-wafer and reconstituted wafer techniques [16]. This approach is an attractive way to integrate two or more incompatible fabrication technologies, e.g. a low-noise RF tier with a high-speed digital tier. It can provide obfuscation if the circuit tiers are fabricated in different foundries, and there has been some work on partitioning guidelines and quantitative metrics of effectiveness [17]. There is opportunity for further work in this area, but we chose not to pursue 3D integration due to resource constraints on the current phase of this project.

The most significant hindrance to the split fabrication approaches is that relevant design tools are, if available at all, primitive and immature. This concern has eventually limited our use of split fabrication in our design.

D. Logic Obfuscation

Logic obfuscation is popular in the literature, e.g., [10][11]. In these active obfuscation techniques, the netlist for the design is altered to obfuscate the logic and to accommodate special circuitry that activates the logic function of the device. The circuit is activated using some type of firmware that is installed in a trusted environment after IC fabrication.

Prior to activation, the chip is resilient against reverse engineering, since the chip is "incomplete" without the necessary activation firmware. Active obfuscation also removes the economic incentive for overproduction, since the chips will not function without activation. Continued protection of the design in the field requires protecting the firmware used in the activation process.

As we explained in Section III.A, we are less concerned with revealing the internal logic of the kernels, since these represent textbook designs. Furthermore, there is a lack of mature CAD tools that can obfuscate the circuit and evaluate the efficacy of the logic obfuscation. For these reasons, we decided not to implement logic obfuscation in this system. As the tool infrastructure matures, we will revisit this decision. We did leverage the concept of circuit activation as a means of protection, which we now describe in the next section.

E. Configurability and Programmability

The benefits and costs of implementing function obfuscation via circuit configurability and programmability are exemplified in the FPGA and GPP. At fabrication time, an FPGA or a GPP effectively hides the in-system behavior of the chip since the function of the part is controlled by the embedded software/firmware, which is installed after fabrication. In addition, they support updating in the field. Configurability and programmability, as illustrated in Figure 3, are the dominant obfuscation techniques used in our design to ensure the security of the design, system capabilities, and algorithms.

The configurability and programmability aspects of our design mitigate concerns of overproduction for gray/black market sales. A chip is of limited value without its customization firmware and instruction system architecture documentation. Indeed, adversaries may consider the uncommitted processor architecture to be a textbook case of SoC architecture, in that case developing anew seems to be a more attractive path than reverse engineering.

The software and firmware, which give the processor its personality, must be protected from reverse engineering in the field. A suitable security architecture that uses a coprocessor and PUF has been developed [2].

F. Implementation

This domain-specific processor was designed to support a broad range of development projects and is protected with three obfuscation techniques: split-fabrication, configurability, and programmability. In addition to this base implementation, which we will refer to as the regular implementation, we have also designed a variant of the processor dedicated for prototyping. The variant will not include split fabrication in order to increase its availability and usability. In this case, instead of using the back-end metal process to establish the connectivity between kernels, the routing switches are designed to be controlled by firmware loaded after fabrication.

The two implementations and their obfuscation techniques are summarized in Table 1, which also includes an FPGA as a baseline in the following discussion.

Table 1: Domain-specific processor implementations for fabrication security and trust (S: split fabrication, C:

configurability, P: programmability, μP: microprocessor, DSP: digital signal processor)

Implementation	Lowest Level Elements	S	C	P
FPGA (Baseline)	LUTs, μPs, DSPs, switch fabrics, memories		X	X
Domain-Specific Processor (Regular)	Domain optimized kernels (FFTs, FIRs, etc.), μPs, switch fabrics, memories, security coprocessors, PUFs	X	X	X
Domain-Specific Processor (Variant for Prototyping)			X	X

In both cases, fabrication security and trust come from the fact that the exact processor architecture remains undefined until post fabrication. All kernels are designed to be configurable by an application-specific bit stream, which defines their parameters (e.g., FFT size and FIR coefficients). The connectivity between the required kernels is also customizable by a configurable switch fabric. Note that an application may not require the participation of all kernels. Furthermore, the data flow and operating steps are controlled by software in the programmable processors, which is loaded only after fabrication.

We have also made design choices that potentially can enhance security and trust at fabrication. For example, as shown in Figure 6, in the design of an FFT kernel, we have created an array of 4K FFT modules and used configurable switch arrays (SWs) to customize them into the sizes required by the operation. This design is not optimal; the FFT sizes in later stages could be made much smaller. This architectural choice increases circuit area, but only incurs a minimal penalty on performance. It has the advantages of modularity and configurability. The fact that the design can be configured into a 1M point sparse FFT or several different regular FFTs not only increases usability, but also helps with obfuscation. We will further discuss the relation between configurability and obfuscation in Section IV.B.

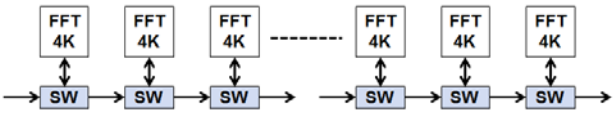

Figure 6: Architecture choice to increase configurability for the FFT kernel (SW: switch array)

IV. Assessment

A. Performance

Table 2 summarizes our assessment on the relative performance merits of the two implementation options using FPGAs as a baseline.

Table 2: Estimated relative performance merits of domain-specific processor implementations comparing to an FPGA (GOPS: billion operations per second)

Implementation	Application-Specific Fabrication Latency	Relative Performance Merits	Remarks
FPGA (Baseline)	None	Power (1X); Throughput (1X); Cost (1X)	E.g., 10 W, 10 GOPs, $5K/chip
Domain-Specific Processor (Regular)	~10 weeks for upper metallization	Power (0.1X); Throughput (10-100X); Cost (40X for 1 wafer)	Cost may reduce to 10X when split fabrication has matured
Domain-Specific Processor (Variant for Prototyping)	None	Power (0.5X); Throughput (10X); Cost (10X for 20 chips)	No special processing requirements

B. Security and Trust

When we set up the threat model, we mentioned that protecting the design logic circuits is not a priority. The manufacturer, by reverse engineering, can determine the kernel function type (e.g., an FFT) from the GDSII netlist. The security and trust are established on the assumption that the kernel parameters (e.g., beyond its maximum FFT size), the connectivity of kernels, and the algorithm for the application remain undetectable by the manufacturer.

In order to quantitatively examine the effectiveness of our design, we have introduced a quantitative metric, based on Shannon entropy [18], to reflect the complexity of determining target application capability and secret sauce by reverse engineering. Our metric is an extension of the cell-level metrics suggested in [19] to measure a design's effectiveness in countering functional and physical profiling. The assessment is performed at the processor's building blocks (i.e., kernels, switch arrays, fabrics, etc.) by determining the number of their possible variations. We define the entropy of a block k as

$$E_k = log_2(number\ of\ k's\ variations)\ bits.$$

We believe that a building block with large entropy is harder for an attacker to determine its use in an application. For example, the function of a fixed kernel design ($E = 0$) can be unambiguously determined. In contrast, it is more challenging to guess how a highly configurable kernel, which has a large E, will be used in an application. We have applied this metric to our assessment of the domain-specific processor in Table 3.

Table 3: Entropies of a domain-specific processor and its building blocks

Entity	Variables	Entropy
FIR	20 32-bit taps	640
FFT	8 FFT settings	3
PPF	100 32-bit taps	3,200
PP	Non-configurable	0

QR	15 matrix sizes	3.9
Fabric	5 by 5 Fully connected	6.9
Processor	All of above	3,853.8

The quantitative relation between entropy and reverse engineering complexity warrants further research, however, we suggest that a design should aim at achieving high entropy. For this paper, we have provided a qualitative assessment of the effectiveness of obfuscation techniques against potential threats, see Table 4.

Table 4: Effectiveness in mitigating fabrication threats

Effectiveness Protecting Against	Regular	Variant
Revealing Application Capability	High	High
Revealing IP: Algorithm	High	High
Revealing IP: Logic	None	None
Function Modification	Medium	Medium
Reliability Reduction	None	None
Repurposing	Medium	Medium

C. Residual Vulnerabilities

Table 4 also exposes the residual vulnerabilities of a domain-specific ASIC processor during an untrusted fabrication. Obfuscation by itself does not prevent unauthorized modification. This is not a unique ASIC vulnerability as both FPGA and GPP need to deal with the same threat. The prevention and detection of Trojan insertions are beyond the scope of this paper, but we believe that, without detailed knowledge of how its target application works, the effort needed for application-specific Trojan insertion increases.

Reproduction for repurposing is definitely possible, but will require a significant reverse engineering effort to determine how to target the "reproduced" processor to an application. As discussed above, we believe that it is more attractive to build a processor anew than to repurpose one by reverse engineering.

V. CONCLUSION

This paper describes our experience designing a domain-specific ASIC processor with obfuscation techniques to protect application information from leaking during fabrication. While obfuscation has been a popular research area, we felt that the lack of mature tools and practical assessment metrics is a major hindrance with using them.

Designers have long accepted the principle of hardware/software co-design. A natural next step is to expand the co-design methodology and include security and trust. This requires a new generation of design tools and metrics to assess the efficacy of obfuscation techniques in meeting mission and application needs.

Last but not least, the traditionally used generic obfuscation benchmarks should be augmented by architecting a few non-obfuscated domain-relevant strawman designs as baselines. These design baselines should facilitate a better understanding of obfuscation techniques in applications and missions.

REFERENCES

[1] D. Whelihan et al, "SHAMROCK: a synthesizable high assurance cryptography and key management coprocessor," MILCOM 2016.

[2] M. Vai et al, "Secure architecture for embedded systems," IEEE HPEC, 2015.

[3] R. P. Cocchi, J. P. Baukus, L. W. Chow and B. J. Wang, "Circuit camouflage integration for hardware IP protection," 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-5.

[4] J. Rajendran, et al, "Security analysis of integrated circuit camouflaging," CCS '13 Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 709-720.

[5] http://ddd.fit.cvut.cz/prj/Benchmarks/, accessed September 2016.

[6] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," SODA, January 2012.

[7] K. Gettings, M. Burke, J. Muldavin, and M. Vai, "Coarse-grain reconfigurable ASIC through multiplexer based switches," IEEE HPEC, 2015.

[8] IARPA Trusted Integrated Circuits BAA, https://www.iarpa.gov/index.php/research-programs/tic/baa, accessed September 2016.

[9] J. Rajendran, O. Sinanoglu and R. Karri, "Is split manufacturing secure," IEEE DATE, 2013.

[10] J. A. Roy, F. Koushanfar and I. L. Markov, "Ending piracy of integrated circuits," in Computer, vol. 43, no. 10, pp. 30-38, Oct. 2010.

[11] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 10, pp. 1493-1502, Oct. 2009.

[12] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri, "Security analysis of logic obfuscation," Design Automation Conference (DAC), 2012, pp. 83-89.

[13] F. Imeson, A. Emtenan, S. Garg and M. Tripunitara, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," Proceedings of the 22nd USENIX Conference on Security (SEC'13), pp. 495-510.

[14] J. Michailos et al., "New challenges and opportunities for 3D integrations," 2015 IEEE International Electron Devices Meeting (IEDM), pp. 8.5.1-8.5.4.

[15] C. Maxfield, "2D vs. 2.5D vs. 3D ICs 101," EE Times, 4/8/2012.

[16] M. Brunnbauer et al., "Embedded wafer level ball grid array (eWLB)," 2008 IEEE Electronics Manufacturing Technology Symposium (IEMT), pp. 1-6.

[17] F. Imeson et al., "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," 22nd USENIX Security Symposium, pp. 495-510.

[18] C. Shannon, "A mathematical theory of communication," Bell System Technical Journal, pp. 379-423, July-October 1948.

[19] M. Jagasivamni et al., "Split-fabrication obfuscation: metrics and techniques," pp. 7-12, IEEE HOST, 2014.