Vertical IP Protection of the Next-Generation Devices: Quo Vadis?

Shubham Rai[1], Siddharth Garg[2], Christian Pilato[3], Vladimir Herdt[4,5], Elmira Moussavi[6], Dominik Sisejkovic[6], Ramesh Karri[2], Rolf Drechsler[4,5], Farhad Merchant[6], Akash Kumar[1]
[1]Chair for Processor Design, TU Dresden, Germany, [2]Center for Cybersecurity, New York University, USA,
[3]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy,
[4]Institute of Computer Science, Univ. of Bremen, Germany, [5]Cyber-Physical Systems, DFKI GmbH, Germany,
[6]Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Germany

***Abstract*—With the advent of 5G and IoT applications, there is a greater thrust in terms of hardware security due to imminent risks caused by high amount of intercommunication between various subsystems. Security gaps in integrated circuits, thus represent high risks for both—the manufacturers and the users of electronic systems. Particularly in the domain of Intellectual Property (IP) protection, there is an urgent need to devise security measures at all levels of abstraction so that we can be one step ahead of any kind of adversarial attacks. This work presents IP protection measures from multiple perspectives—from system-level down to device-level security measures, from discussing various attack methods such as reverse engineering and hardware Trojan insertions to proposing new-age protection measures such as multi-valued logic locking and secure information flow tracking. This special session will give a holistic overview at the current state-of-the-art measures and how well we are prepared for the next generation circuits and systems.**

I. Introduction

Security has emerged as an equally important metric along with area, power, and delay while designing ASICs [1]. Designers worldwide see the need to incorporate security measures in their designs and verification; hence, adding security benefits is a welcome cost. However, with the globalization of the supply chain for electronic circuits, giving security guarantees with CMOS-based circuits often comes with huge area and performance overheads [2]. On top of that, this distributed setup of fabrication, foundry and testing allows several points where adversaries can attack the chip design or make illicit copies. Hence, various attacks such as inclusion of a particular hardware Trojans, IP piracy, IC overbuilding, reverse engineering, IC counterfeiting and side-channel attacks are prevalent which can be detrimental for both the design-houses as well as the end-user. While the design-houses face the issue of infringement of their intellectual properties (IP), the end-user faces the ultimate risk of losing secret or private information. Researchers around the world have worked on various countermeasures like logic encryption, IP watermarking, IC camouflaging etc. to cope with such attacks [3].

With the growing proliferation of electronic circuits owing to the demand of autonomous vehicles and IoT applications, hardware security needs to be ensured at all levels of abstraction [4]. Hence, in this work, we tackle the challenges of IP protection with a *vertical* approach – from system to device level. We first discuss system-level IP protection. Section II details obfuscation techniques for semantics-aware IP protection. Section III discusses how information flow tracking can be used for security validation at *Virtual Prototype* (VP) level. We then describe techniques which are possible at lower abstractions. While Section IV introduces multi-valued technique for future generation computing systems, Section V describes an emerging reconfigurable nanotechnology which allows building of polymorphic circuits from bottom-up. At each level, we describe the techniques, present the current results, but also pose ourselves the research question *Quo vadis?*, reasoning about the future trends.

II. RTL Obfuscation of Semantics of IP

Next-generation integrated circuit (IC) designs will be increasingly composed of off-the-shelf components (COTS) and specialized intellectual property (IP) blocks. While COTS represent pre-existing components (e.g., processors, memory controllers, etc.), the specialized IP blocks often represent the added value of the product. For example, they can implement proprietary algorithms for digital signal processing or data analytics. Due to the increasing cost of IC manufacturing, design houses are forced to outsource the fabrication to a third-party foundry, exposing the IP to security vulnerabilities. Protecting the IP of these IC components is vital not only for the economy of the company but also for the safety and security of the end product. If malicious attackers are able to reverse engineer the IP functionality, they can replicate and re-sell the IC at a lower cost. They can introduce malicious modifications to harm the normal execution of the legitimate ICs. However, integrating security protections must fit into the existing/established industrial design flows to avoid compromising the time-to-market and quality of the final ICs without affecting the security guarantees. For example, high-level obfuscation can be performed during high-level synthesis but requires custom tools [5].

In this section we present a new direction in obfuscation to protect the semantics of an IP against an untrusted foundry to fit into existing EDA flows[1]. We present ASSURE [6] approach that operates at the register-transfer level (RTL) to obfuscate behavioral IP.

A. ASSURE RTL Obfuscation

ASSURE is compatible with industrial design flows, as shown in Fig. 1, and assumes the attackers are in the foundry. They have access to an obfuscated netlist. A functional IC – *oracle* – is unavailable (*oracle-less* model). This is a legitimate threat

[1]Not requiring any changes to the EDA flows.

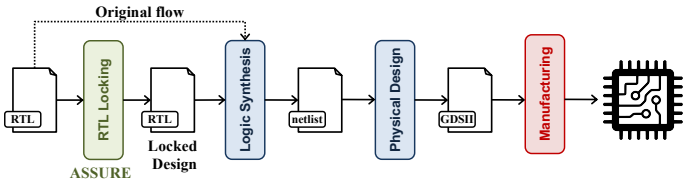


Fig. 1: IC design flow with ASSURE RTL Obfuscation. ASSURE provides semantic-aware IP protection against an untrusted foundry.

model for low-volume ICs, like those used in defense systems, where a working IC is unavailable to the attackers.

Given a behavioral RTL design, ASSURE produces a obfuscated version and the corresponding bitstream to make it functional post-fabrication. ASSURE obfuscates semantic elements like constants, operations, and branches with *opaque predicates* as in *software obfuscation* [7]. ASSURE opaque predicates depend on the locking bitstream, which is known during obfuscation but unknown during attack. Fig. 2 shows the complete process. We discuss two representative designs, AES and DFT from the MIT-LL Common Evaluation Platform[2] (CEP) to motivate our conclusions on RTL obfuscation. AES requires a total of 819,726 bits for obfuscation, mostly due to the S-box constants. Obfuscating these elements is interesting in case of a secret S-box. For example, there are alternative s-boxes and a nation-state may use one from among those and redact them. Discrete Fourier Transform (DFT) requires, instead, 8,697 bits distributed across the three types.

For each input design, we generated several variants obtained by activating different obfuscation techniques and providing different key budgets. ASSURE operates directly on *abstract syntax tree* (AST) of input HDL description and is so independent of the input flow. First, we determine which elements to protect. Obfuscating all elements may not exceed the numbers of bits available in the tamper-proof memory (TPM). For the table-based implementation of AES, for instance, all 819,726 bits can be obfuscated. DFT can be entirely obfuscated with a large TPM. ASSURE performs a *depth-first analysis* and obfuscates elements as long as there there are sufficient bits. This method generates always feasible designs, letting the designer decide how many bits to use. Once the elements are identified, ASSURE applies obfuscation by manipulating the AST. Fig. 2 shows the AST manipulation to extract one constant of the AES or obfuscate an operation. Branches are obfuscated with XOR gates on the predicates to disguise the identification of the `true`/`false` blocks. The resulting AST enters into the *RTL generation* phase that implements a Verilog backend. The output RTL design has the same top interface as the original module, except for an additional input port to deliver the locking bitstream.

ASSURE is an RTL-Verilog-In-RTL-Verilog-Out tool that starts with a synthesizable RTL verilog IP without any additional constraints. The output RTL can replace the original RTL in any existing EDA flow, including during the simulation and verification steps. We can formally verify the obfuscated design by matching the input RTL and the output RTL when the correct key bitstream is loaded. We verify that any other bitstream

[2]https://github.com/mit-ll/CEP

TABLE I: Overhead analysis for the AES and DFT designs using different key bitstream sizes (Nangate 15nm).

Design	Bits	Area	Power	Critical Path
Original AES	-	42,854.69 μm^2	4.47 mW	136.75 ns
Obf. only consts	819,296	+446%	+261%	-29%
Obf. only ops	429	+6%	+18%	+2%
Obf. only branches	1	=	+8%	-1%
Obf. all elements	819,726	+499%	+274%	+6%
Original DFT	-	81,865.94 μm^2	10.46 mW	336.72 ns
Obf. only consts	8,414	+4%	+3%	-4%
Obf. only ops	151	+1%	+2%	+13%
Obf. only branches	132	=	+2%	-4%
Obf. all elements	8,697	+7%	+8%	+18%

introduces at least one failing point in the design during the verification, proving that the obfuscated functionality is not activated with a different bitstream. The obfuscated RTL can be the golden reference in the rest of the IC design flow.

B. Security Analysis

ASSURE techniques offer provable security guarantees against oracle-less attacks [6, 8]. Any opaque predicate we generate during obfuscation involves a specific portion of the input key bitstream. ASSURE generates a obfuscated circuit and a bitstream $\mathcal{K}^*$. The circuit is *indistinguishable* from the ones generated with any other $\mathcal{K}_i \neq \mathcal{K}^*$ when the attacker has no prior information on the design. We performed an experimental evaluation of these effects by applying correct and incorrect bitstreams, and matching the resulting designs with the original one. On one hand, obfuscated designs always match the original ones when using the correct bitstreams. On the other hand, incorrect bitstream always introduce at least one failing point in equivalence checking.

C. Power, Performance, and Area Overheads

We synthesized the designs with Synopsys Design Compiler J-2018.04-SP5 on the Nangate 15nm technology at standard operating conditions. We evaluated the impact of obfuscating constants, operations, and branches separately, as well as all together. Table I shows area, power, and delay overheads compared to the corresponding original, unprotected versions. While timing effects mostly depends on where the obfuscation is applied, area overhead is proportional to the size of the key bitstream used for obfuscation. Obfuscating the constants requires most of the key bitstream (e.e., 99% and 96% for AES and DFT, respectively). However, the impact per bit is lower than in operation obfuscation. Indeed, operation obfuscation requires additional hardware resources and so is more expensive. On the contrary, obfuscating constants do not introduce extra logic but prevents logic-level optimizations like constant propagation. Branch obfuscation is inexpensive in terms of area, but introduces extra power consumption.

D. Discussion and Future Work

ASSURE is a provably-secure RTL obfuscation scheme. Raising the abstraction level has multiple advantages: 1) it allows us to protect the IP semantics before it is optimized

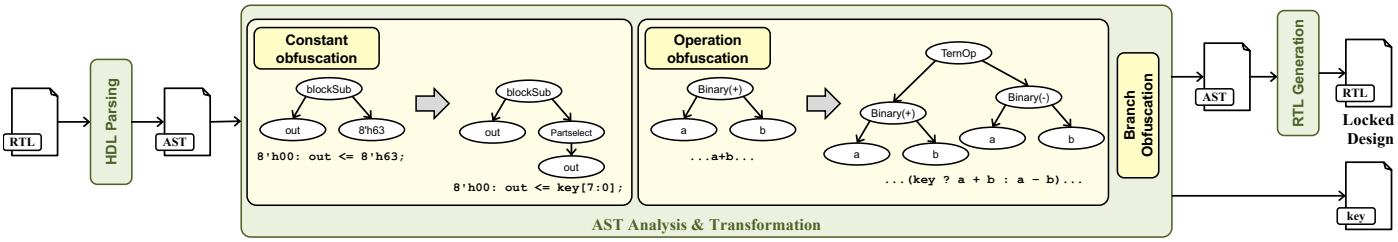


Fig. 2: ASSURE obfuscation flow.

and embedded into the netlist by logic synthesis, and 2) it is compatible with industrial EDA flows, making it valuable for semiconductor design houses. Our next steps are:

- validate ASSURE on larger designs and with more structural and functional metrics and attacks. We plan to use a red team-blue team approach to create a virtuous cycle of attacks and defenses [5, 9].
- include more obfuscation techniques, borrowing concepts from software obfuscation, and methods to automatically select which portions of the design to obfuscate with limited key bitstream budgets and/or overhead constraints.
- extend the approach to thwart also oracle-guided, SAT-based attacks. The extension demands methods to make SAT instances and, in turn, key bitstream recovery exponentially more complex if not impossible.

III. Security validation at VP-level using information flow tracking

Security is a crucial aspect in modern embedded systems that complements functional correctness to build safe and reliable systems. A very effective technique to validate security policies and thus protect a system against a broad range of security related exploits is *Information Flow Tracking* (IFT) [10, 11].

In this section we present efficient IFT-based techniques at the system-level using *Virtual Prototypes* (VPs). This allows validation of security policies early in the design flow and hence enables to prevent costly iterations later on. We present static and dynamic IFT-based techniques for security validation of the VP as well as the embedded SW running on the VP. Our experiments demonstrate the effectiveness of our approach.

In the following, we start with an overview of VP-based IFT, then summarize our experimental results and discuss the open challenges and opportunities for future work in this domain.

A. Overview

Fig. 3 shows an overview on our VP-based IFT approaches. Starting point is a security policy that needs to be validated at the VP level. A security policy enables to specify fine grained access control models, in particular to cover confidentiality and integrity aspects from an execution perspective. Confidentiality ensures that secret data (provided in secret memory or register locations) does not leak to untrusted targets. Integrity ensures that untrusted data (in particular coming through untrusted input interfaces) does not affect sensitive data. IFT enables to protect against a broad range of security related exploits (e.g. which affect confidentiality or integrity) by tracking and checking the information flow alongside the software execution. We consider

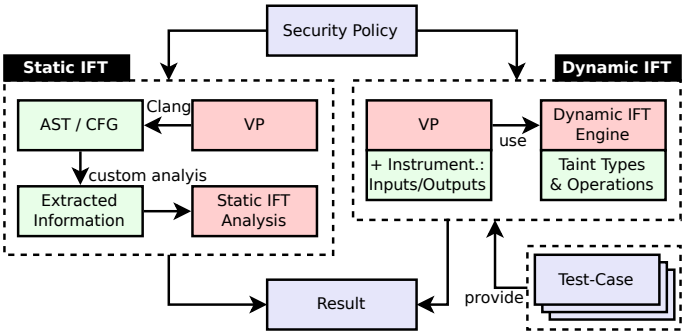


Fig. 3: VP-based static and dynamic IFT for security policy validation.

static and dynamic IFT for validation of security policies. Fig. 3 shows an overview on both approaches (static IFT on left side and dynamic IFT on right side).

1) Static IFT: Our static IFT approach essentially works in three stages [12]. First the VP modules are transformed into *Abstract Syntax Tree* (AST) and *Control Flow Graph* (CFG) representations using the *Clang* compiler. In the second step multiple static analysis techniques are performed on these intermediate representations to extract relevant information for the subsequent IFT analysis step. This includes binding information, call graphs and data flow relations such as def-use pairs (i.e. connection a variable definition with it's use). Based on this local information a global IFT analysis is performed that essentially propagates data flow information between modules and from different inputs to outputs. We perform a conservative analysis, which considers all static paths in the VP, hence the analysis is sound but may produce false positives by finding spurious information flows which may never occur at runtime.

2) Dynamic IFT: Dynamic IFT essentially works by tracking information flow at runtime alongside the normal VP execution [13]. The VP is instrumented to tag all registers and memory locations with data flow information. These tags are initialized at input devices or at pre-defined memory locations. During execution the tags are propagated according to the security policy. This requires instrumentation of the instruction set simulator and the peripherals. A dynamic IFT engine provides data types and operations to implement the tagging. By leveraging C++ operator overloading and templates, a mostly non-intrusive integration is achieved, which is compatible with the TLM 2.0 communication standard. The propagated tags are checked at runtime to detect and prevent security policy violations. A comprehensive evaluation requires a set of test-cases which provide the VP input stimuli.

B. Experimental Results

We now summarize the main results obtained by our VP-based static and dynamic IFT, from [12] and [13], respectively.

1) Static IFT: We evaluated this approach using the LEON3 based SoCRocket VP [14] which is available at GitHub [15]. The VP integrates several components around an AHB/APB AMBA-2.0 bus system. As part of the evaluation we considered the integration of different additional TLM IP components.

In particular, we investigated a Crypto AES IP which is a hardware accelerator for the AES-128 algorithm, an NFC interface IP for near field communication of two devices in close proximity, and a smart card reader IP which reads data from a card and stores it into a secure location. The VP-based static IFT approach has been very effective in finding intricate security policy violations. This included access to secret data through an open debug interface or using DMA to bypass the normal memory access. The approach also works very efficiently requiring around 50 to 75 seconds, depending on the IP, to construct and analyze all available static paths. Since the analysis is conservative, it is able to prove that a specific data flow is indeed not possible.

2) Dynamic IFT: We have implemented our dynamic IFT approach for security policy evaluation on top of the open source RISC-V VP [16, 17] available at GitHub [18] that is implemented in SystemC TLM. For the evaluation we considered three different experiments.

In the first experiment we considered an ECU of a car engine immobilizer as a case-study. The immobilizer holds a secret PIN in memory for authentication purposes with the ECU by means of a challenge/response protocol. For encryption purposes the AES protocol is utilized. The security policy is to ensure that the secret PIN is neither leaked (which would be a confidentiality violation) nor modified by unauthorized access (which would be an integrity violation). In our VP-based evaluation we demonstrated that several common attack scenarios are detected and prevented by the dynamic IFT approach. Our manually performed attacks included buffer overflows, attempts to override the PIN with external data and using the PIN in control flow statements (which could leak it indirectly). Our approach is very beneficial for early validation of security policies.

In the second experiment we demonstrated that a dynamic IFT approach enables strong protection against code injection attacks. For evaluation purposes we used the Wilander-Kamkar buffer overflow attack suite [19] which has been ported to RISC-V by [20]. It includes several attack pattern to achieve code injection or remote code execution by triggering a buffer overflow on stack or heap regions. Such a buffer overflow can for example be used to overwrite the return pointer address. With an appropriate security policy, all attack scenarios have been detected and thus prevented.

The third and last experiment measured the performance overhead added by the dynamic IFT engine to the normal VP-based execution. Fig. 4 shows the results on seven benchmarks that range from pure CPU benchmarks (*qsort*, *dhrystone*, *primes*, *sha512*) to full system benchmarks (*simple-sensor*, *immo-fixed*) and operating systems (*freertos-tasks*). The left side compares the execution time in seconds and the right side the obtained MIPS (*Million Instructions Per Second*) for the normal VP (blue color) and the VP with dynamic IFT integration (orange color). On average they achieve 33.2 MIPS and 17.0 MIPS, respectively, which corresponds to a performance overhead of around 2x.

Compared to static IFT, false positives are avoided by tracking precise runtime information, however, dynamic IFT relies on test-cases to achieve comprehensive evaluation results.

C. Discussion and Future Work

VP-based IFT enables early validation of security policies to protect against a broad range of security attacks. Static and dynamic IFT have been shown very effective and complementary in this regard. To further improve them, we plan to:

- Consider automated test generation techniques that are tailored for validation of security policies to boost the dynamic IFT approach further. Techniques based on fuzzing and symbolic execution, e.g. the VP-based SW verification techniques [21, 22, 23], seem promising. Modern fuzzing-based approaches are guided by code coverage and employ randomized mutations. It would be interesting to investigate feedback methods that consider data flow relations to boost the test generation process with respect to security policies. Concolic testing is another promising technique that tracks constraints alongside the program execution to continuously generate new inputs to drive the execution towards new paths. A specialized exploration strategy can improve the bug hunting capabilities significantly.
- Investigate more precise static analysis techniques for the static IFT approach to further reduce the detection of spurious security policy violations by providing less conservative results. A complementary direction is to investigate compiler extension techniques to annotate information to the software program that enable to provide accurate results without over-approximation. Conceptually, this follows the idea of safe C dialects that offer language extensions to provide a framework that enables efficient protection against certain error classes such as buffer overflows.
- Devise a unified framework that efficiently combines static and dynamic IFT at the VP level. Such an approach would introduce benefits of both techniques: a sound and complete analysis as offered by static IFT and a precise analysis based on runtime information as offered by dynamic IFT. Moreover, as an intermediate step, both approaches are complementary and can benefit from each other. Dynamic IFT strongly relies on good test generation methods which can significantly benefit from available static information. Static IFT can utilize runtime information collected for specific paths to make the analysis techniques more precise.

IV. MVLock: A multi-valued logic locking scheme for future-generation computing systems

The future-generation computing systems will require sophisticated security mechanisms to prevent a variety of attacks. Especially with the emergence of neuromorphic computing, the underlying computations are not purely digital anymore. The complexity of the future-generation computing systems also

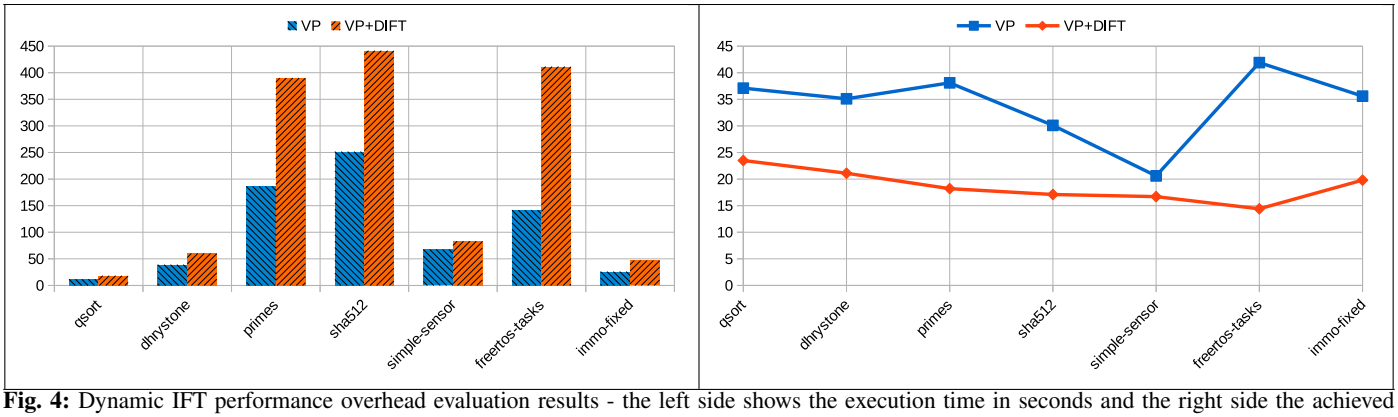


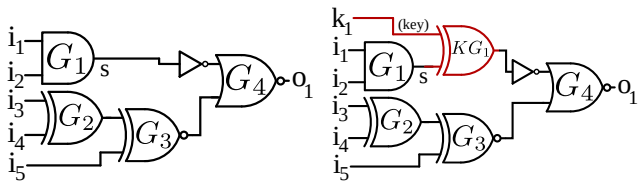
Fig. 4: Dynamic IFT performance overhead evaluation results - the left side shows the execution time in seconds and the right side the achieved MIPS for the normal VP (blue color) and the VP with dynamic IFT integration (orange color).

increases the attack surface for the bad actors. The vulnerability of the designs while in the production at a third-party foundry is going to be a major concern. Logic locking is an emerging technique able to provide various measures to protect against foundry attacks such as hardware Trojan insertion, IP piracy, and counterfeiting. In this section, we discuss the impact of post-CMOS technologies on security and how various logic locking paradigms can help us overcome hardware security challenges. Here, in particular, we propose integrating soft (biological) intelligent systems as high-density building-blocks to store information and create a subset of multi-valued logic locking (MVLock), and discuss increasing difficulty level in breaking the logic locked circuits. Classical Boolean satisfiability test-based attacks and novel machine learning based attacks are analysed for key retrieval and prediction.

Due to the prohibitive cost and complexity of constructing and maintaining a semiconductor foundry with high capability in fabrication, most integrated circuit (IC) design houses are becoming fabless. Moreover, the importance of time-to-market is also compelling the IC design companies to rely on the third party IC intellectual property (IP) blocks and utilizing them in their system-on-chip and outsourcing the fabrication to advanced offshore foundries. The globalization of IC fabrication supply chain has raised risk of various kinds of adversarial attacks ranging from IP piracy to hardware Trojans [24].

A. CMOS-based Logic Locking

Logic locking performs a functional and structural manipulation of a hardware design through the insertion of additional obfuscation logic, thereby creating a dependency on a secret key [25]. If a correct activation key is provided, the design behaves as expected. Otherwise, an incorrect key ensures the



(a) Original IC **(b)** Locked IC

Fig. 5: Example: XOR/XNOR-based logic locking

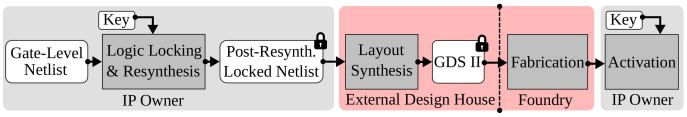


Fig. 6: Logic locking in the IC design flow

generation of faulty outputs for at least some input patterns. This process is typically performed on a gate-level netlist. One of the first logic locking schemes is known as EPIC [26]. This scheme randomly disseminates XOR/XNOR gates in the netlist. These gates are typically known as *key gates* (KGs). To understand how EPIC operates, let us consider the example in Fig. 5. Here, the original circuit is locked through the insertion of a single XOR key gate KG_1 driven by the key input k_1. If $k_1 = 0$, the value of the output s of gate G_1 is preserved. Otherwise, the value of s is inverted, effectively corrupting the functionality of the netlist. The same is true for an XNOR gate and a key of value 1. Note that the security of this scheme relies on the fact that a simple removal of the key gates is not possible, since the attacker cannot distinguish whether an inverter is part of the original circuit or the key gate. In the past years, a wide range of locking schemes has been proposed, thereby operating with XOR, XNOR, AND, OR, MUX gates or more complex structures [27, 28].

Logic locking plays a vital role in the IC design and fabrication flow [29, 30, 31]. As depicted in Fig. 6, first, the IP owner (trusted regime) deploys logic locking on a gate-level netlist and performs another logic synthesis round. The resynthesis is often a crucial step to further integrate any changes induced by the locking mechanism. Afterwards, the locked netlist proceeds into the layout generation stage which can be performed in-house (trusted) or outsourced to a third-party design house (untrusted). In both scenarios, the generated layout is dispatched to a foundry (untrusted). All untrusted parties can potentially steal the IP or insert malicious modifications into the design before fabrication. To overcome this vulnerability, logic locking binds the design to a secret key which is only known to the IP owner. Therefore, the IP is concealed throughout the untrusted regime, as it forces the untrusted parties to first recover the key before being able to reverse engineer, understand and intelligible modify the IP.

TABLE II: Signed-Ternary Multi-Value Logic

A	-1	0	+1
not A	+1	0	-1

B. Post-CMOS Technologies for Logic Locking

With the advent of post-CMOS devices such as nanowire transistors, carbon based, spin-based devices, smaller electronics with lower power overheads can be achieved as compared to CMOS counterparts [32]. In addition, these emerging devices can improve hardware security based on the aspect of polymorphism. A polymorphic logic gate is able to perform distinct Boolean logic functions, i.e., AND/NAND, OR/NOR, XOR/XNOR, by configuring internal/external keys at the run-time. Recently, polymorphic logic gates have played a crucial role in addressing IC-related security issues, including counterfeiting and reverse engineering, as well as supporting camouflaging and locking [33]. Firstly, having a uniform device-level layout can make it harder to determine the functionality, specifically for optical-imaging-based reverse engineering. Secondly, the intrinsic functionality of a polymorphic gate is dependent on the control key input [34].

Memristors are emerging electronic devices which have two-terminal resistive switches and can improve security by leveraging the unique properties of memristors [35]. Properties of memristors such as non-volatility, fast switching behavior, nanoscale dimensions, CMOS compatibility and low power consumption present new opportunities for realizing ultra high-density memory arrays and building security primitives. Memristive devices enable the integration of security, memory and computing functionalities into the same circuits based on the inherent reconfigurability and variability of the memristors [36]. Polymorphic electronics are introduced based on the idea of having multiple functionalities built in the same cell, controlling the input-output relation in the circuit to hide the original design functionality in the form of hardware obfuscation.

Obfuscation techniques can also involve camouflaged cells to increase the effort needed by an attacker to reverse engineer the logic by determining the functionality of the cell from its layout or introduce additional gates to lock the proper functionality of the protected circuit. However, it still remains a challenge to achieve immunity against reverse engineering, especially in the presence of IC imaging methods [37]. To address these challenges, there is a growing interest to leverage the intrinsic characteristics of the transistors in order to create camouflaged gates, e.g., by leveraging multi-threshold-voltage transistors for design obfuscation. Therefore, multi-valued logic gates (MVLGs) introduce the aspect of polymorphism to circuit inputs/outputs by leveraging post-CMOS devices. This feature rises the complexity of performing major attacks on the design, where current logic-locking approaches fail at gate level.

C. MVLock

We introduce the advantage of leveraging multi-valued gates to protect a design on circuit level and to support both logic locking and camouflaging. We can improve camouflaging of logic cells by utilizing the same physical structure to implement a large number of different logic functionalities based on a secret key. To implement various logic functions, a truth-table can be generated with all input combinations of the transistors, which ideally has r^{r^n} possible functions, where n is the number of inputs and r is the radix. In other worlds, the logic-locking circuitry can be designed with other logic gates (apart from Boolean logic) in a multi-valued design framework. One example of multi-valued logic for $r = 3$ (ternary) is presented in Table II. The input variable **A** can take the value of -1, 0 or 1. Consequently, the value of **not A** takes the value of 1, 0 and -1. This concept can be ported to locking methodologies using multi-valued logic. Herewith, MVLock brings another level of complexity for the hardware obfuscation and also for the attacker.

In the context of implementing MVLGs, system integrated ion-sensitive field effect transistors (ISFETs) are interesting due to their well-established fabrication and identical sensor characteristics at wafer scale for biodetection [38]. Hereby, silicon nanowires (SiNWs) have been fabricated for biosensing applications based on top-down processed ISFETs. SiNW-arrays exhibit superior sensor characteristics, thereby enabling differential readout and multichannel capabilities [39, 40]. Furthermore, ISFETs are compatible with a CMOS integration [41]. One of the attractive approaches of utilizing non-metalized silicon FET-microarrays has been reported as a method to detect and monitor DNA hybridization, which can enable a fast, fully electronic and stable differential AC readout free of side parameters and detected point-mutations (or nucleotide polymorphisms) of short DNA sequences [42]. The complementary silicon nanowires field effect transistors can be used to develop label free, ultra sensitive biosensor applications, representing bio-nanoelectronics-based logic locking for security systems.

D. Attack Scenarios

The paradigm shift in computational architectures, especially with evolving neuromorphic computation with biochemical reactions, brings new security challenges in designing processors. Interestingly, multi-valued logic operations are likely to become a common feature of next-generation processor architectures. Hereby, the polymorphic characteristic of MVLGs can help to overcome major attacks where current logic locking structures fail. Since traditional logic locking is limited to binary CMOS-based logic, it is open to a variety of key-recovery attacks, including the Boolean satisfiability (SAT) attack [43] and structural analysis attacks utilizing novel machine learning methods [44]. Therefore, the proposed bio-nanoelectronics-based approach using post-CMOS computational building blocks offers a novel approach to hardware integrity protection.

E. Discussion and Future Work

We identified the binary nature of logic key gates as a fundamental limitation of traditional logic locking. Therefore, we propose integrating biologically activated nanoscale field-effect transistors as functional key gates alongside a locked CMOS netlist. By utilizing multi-layer MVLGs in the form of unique biological activation keys, MVLock is a promising approach for protecting the integrity of future-generation circuits against malicious actors in the IC supply chain. Future work directions include:

- Establishing provable security guarantees with MVLock
- Exploring a variety of MVLG-based locking mechanisms as well as novel key-recovery attacks.
- Developing solutions based on emerging technologies to supprt MVLock

V. Harnessing security through runtime reconfigurable Transistors

While the previous sections described various hardware security schemes at system or circuit level, in this section, we look at an emerging nanotechnology which can provide hardware security from the device level.

A. Reconfigurable Nanotechnologies

Ambipolarity or ambipolar conduction is a natural physical phenomenon observed in technology nodes below 45nm where both charge carriers can flow in the channel on application of voltage potential [45]. Hence, transistors based on materials such as silicon or germanium tend to exhibit both p- and n-type of conduction at lower technology nodes. In conventional manufacturing process, one of the charge carriers is intentionally suppressed using dopant concentration to enable only one type of conduction characteristics. However, recently several emerging nanotechnologies based on materials such as silicon [46, 45], germanium [47], graphene [48] and carbon [49] exploit this ambipolarity to enable devices which can exhibit both kind of conduction on application of an external potential. Devices which can exploit this ambipolarity are often termed as *Reconfigurable Field Effect Transistors* (RFETs). Based on their device geometry, RFETs can be broadly classified as 1D devices such as silicon or germanium nanowires or 2D devices based on materials such as graphene [48] or other transition metal dicalchogenide (TMD) materials like $MoTe_2$ [50], WSe_2 [51].

1D RFETs are more mature as compared to 2D devices due to their similarity to CMOS manufacturing process [52, 53, 54]. The stacked nanowire or nanosheet geometry is also considered as a successor to the *FinFET* geometry that is promoted to be used at lower technology nodes [55]. Additionally, silicon and germanium are one of most common materials used in conventional MOSFET technology. Owing to the similarity to CMOS integration process, RFETs are one of the commercially feasible emerging nanotechnologies.

One of the standout features for RFETs which makes them distinct from conventional CMOS, is that transistors based on these nanotechnologies consist of two or more gate terminals to allow runtime reconfiguration between the p- and n-type electrical characteristics. One of the gate types which is called the *Control Gate* (CG) is analogous to the gate terminal in conventional CMOS and controls the flow of charge carriers. The other gate type is called the *Program Gate* (PG) and it controls the type of charge carriers in the channel. The PG plays the major role in *programming* the device to function either as the p-type or n-type. Further details about the device physics of RFETs can be found in [54].

These reconfigurable properties form the very basis, why RFETs are so widely applicable in hardware security [56, 57, 58, 59]. We look at few of the security schemes which are possible due to this inherent reconfiguration.

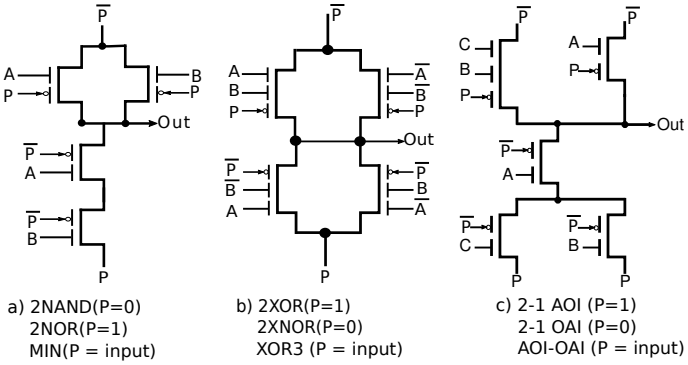


Fig. 7: Reconfigurable logic gates using RFETs [60]

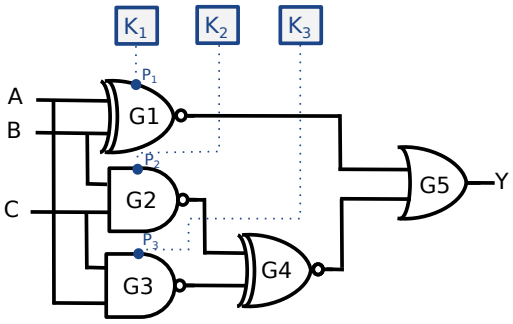


Fig. 8: The program gate (PG) acts as the key-input for RFET-based logic locking. Polymorphic logic gates realizes logic locking without additional of additional logic gates.

B. Polymorphic Logic Gates and Logic Locking

At the logical abstraction, the runtime-reconfigurable properties offered by RFETs can be used to build logic gates with extended functionality [61, 52, 57, 62, 63]. These logic gates can be configured to deliver different logic functionalities on application of an external potential. Some of the logic gates based on RFETs are shown in Fig. 7. We can notice that the functionality of individual logic gates changes depending upon the value of the program gate terminal P. In this direction, it was also shown recently that RFETs are more efficient to implement *Self-Dual* logic functionality with inherent reconfigurability than conventional CMOS [64].

This functional reconfigurability can be used in various IP protection schemes as it allows efficient and cost-effective polymorphic logic gates [65, 66]. These polymorphic logic gates can be used for logic locking schemes [56]. The conventional locking scheme as shown in Fig. 5 can easily be realized without additional logic gates as shown in Fig. 8. Evaluation of any logic-locking scheme using the seminal SAT-based attack [67] is essential to analyze its efficacy. Hence, we evaluated RFET-based logic locked circuit using the SAT-based attacks and found that the RFET-based circuits are capable enough to provide practical security against such SAT-based attacks. We have used ITC-99 benchmarks [68] and show that beyond 30% locking, the SAT-based attacks reach to time-out and were not able to detect the key [56].

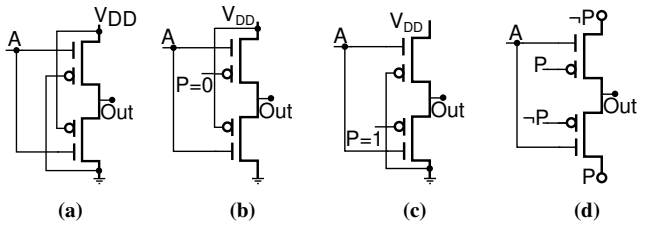


Fig. 9: Connections for inverters [60] (a) Static design. The drain, source and program gate are hard fixed to V_{dd} and V_{ss} (b) One transistor's program terminal is connected to 0 (c) One transistor's program terminal is connected to 1 (d) Fully reconfigurable design.

C. Robustness against side-channel effects

Side-channel attacks using effects such as magnetic effects, temperature effects or electrical effects are potential mechanism generally used during reverse engineering [3]. Circuits exposed to these attacks often lead to leakage of information and other security compromises [69, 70]. Of these, differential power analysis is one of the most common side-channel attack technique which exploits the relationship between input and the output of a particular logic gate. In particular, XOR logic gates which are heavily used in most of the cryptographic domain application has a representative power trace which is easy to recognize during reverse engineering. Additionally, for CMOS devices, the skew in electrical characteristics between p- and n-type (and hence in pull-up and pull-down operation) makes it easier for power differential analysis. One of the effective countermeasures for such differential power analysis is to employ complementary logic operation (such as XOR and XNOR or OR and NOR) that allows mixing of input data patterns which makes it difficult for side-channel attacks to figure out the power trace of the underlying logic operation. In this direction, RFETs can be used to prevent such attacks.

In RFETs-based circuits, the skew between the pull-up and pull-down network for a given circuit has almost vanished. Hence, complementary logic such as XOR-XNOR are much more efficient to realize in RFETs [71, 72, 60]. As discussed, complementary circuits such as Boolean functions which are dual to each other can also be efficiently realized using RFETs. These logic gates can contribute in building circuits based on RFETs that are inherently robust as compared to CMOS-based circuits with minimal area, delay or power overheads [71].

D. The curious case of RFET-based inverters

Due to this inherent possibility to configure individual RFETs, inverters present an interesting case. Multiple variants of RFETs-based inverters can be realized depending upon the connection to the program and control gate of individual transistors. These multiple variants are shown in Fig. 9. Hence, RFETs-based inverters present polymorphism from an altogether different angle, where the functionality remains the same but structurally they are different. Such multiple variants of the same logic gate can be used in watermarking schemes [73] or camouflaging techniques [74].

E. Security vulnerability

While RFETs-based circuit topology blurs the distinction between pull-up and pull-down network to enable logic gates

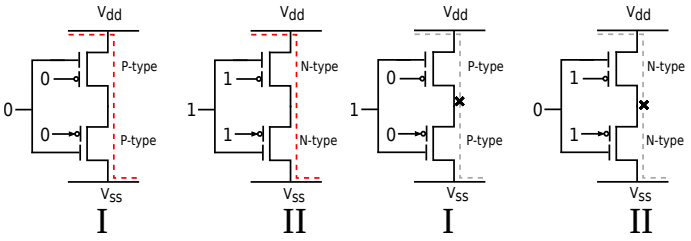


Fig. 10: Short-circuit and open-circuit configurations in RFETs-based inverters.

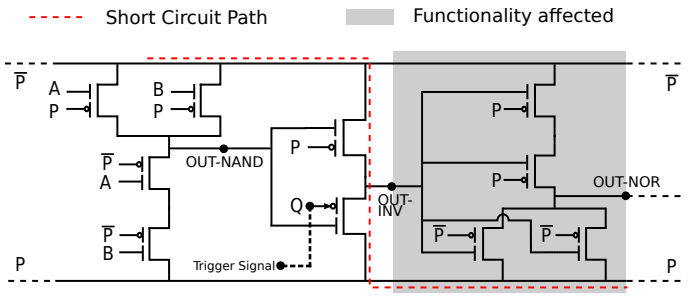


Fig. 11: A sub-circuit consisting of NAND, INVERTER and NOR, where the INVERTER is misconfigured.

and circuits with multiple functionalities, the same feature can be exploited in terms of a potential security vulnerability which can prove detrimental for the circuit functionality. RFETs, just like CMOS work in the same circuit paradigm which requires complementary networks (pull-up and pull-down) to enable logic 1 and logic 0. However, unlike CMOS, individual RFETs (or group of RFETs) can be independently configured to render p-type or n-type behavior. A detrimental scenario arises when one or more RFETs in a logic gate are intentionally misconfigured to disrupt this complementary network. We demonstrate this intentional misconfiguration in case of RFETs-based inverters as shown in Fig. 10. One can notice that two separate circuit configurations – *Open-Circuit* and *Short-circuit* can arise if the gates of an RFETs are not configured correctly. Such open- and short-circuit scenarios can be enabled at any given time in otherwise normal functioning RFETs-based circuit to render it error-prone.

The unique aspect of such RFET-based circuits is that these intentional misconfigurations can be realized in any RFET-based logic gate at any given time. Additionally, such intentional misconfigurations also evade the testing phase because the circuit behaves normally with normal configurations. Only in case of explicit perturbation, such misconfiguration can manifest leading to adversarial scenarios. Such perturbations can be triggered either by some external rare events or some internal faults or aging effects.

We evaluate such intentional misconfigurations in a small subcircuit as shown in Fig. 11. The TABLE III shows the circuit level simulation for the circuit shown in Fig. 11. Intentional misconfigurations are carried in the middle inverter by changing the potential at Q. We can notice, that due to the intentional misconfiguration, both the RFETs of the inverter can be configured as p-type or n-type thereby creating a direct path or an open path. This causes the point *OUT_INV* to be at

TABLE III: Simulation results showing current drawn and voltage values for different cases of inputs and configurations

	Overall Current Drawn through INV			At Node OUT-INV			At final Output		
	Normal Operation	**Both n-type**	**Both p-type**	**Normal Operation**	**Both n-type**	**Both p-type**	**Normal Operation**	**Both n-type**	**Both p-type**
INV-INV-INV									
Output-1 = 0V	389pA	3.05uA	8.19pA	0V	31.13mV	686mV	0V	700mV	0V
Output-1 = 700mV	83pA	251pA	2.5uA	700mV	106.7mV	609mV	700mV	700mV	0V
XOR-INV-NAND									
OUT-XOR = 0V	414 pA	251.8pA	2.5uA	700mV	31.06mV	610 mV	0V	700mV	0V
OUT-XOR = 700mV	11.58pA	3.0 uA	7.17pA	0V	100mV	688mV	700mV	700mV	0V
NAND-INV-NOR									
OUT-NAND = 0V	414pA	251pA	2.5uA	700mV	31mV	609mV	0V	700mV	0V
OUT-NAND = 700mV	11.8pA	3.055uA	29.01pA	0V	106mV	688mV	700mV	700mV	0V

indeterminate potential which disrupts the successive stages of the combinational path. This can be ascertained by looking at the voltages of the final output in TABLE III.

Such disruption in the circuit can lead to derailment of logic values at the output. Additionally, due to short circuit scenarios, large amount of current can be drawn from V_{dd}. This discharge of current can lead to much more adverse reliability effects [56]. These effects have greater repercussions as scenario are far more severe as it can present itself in the form of higher dynamic and static power dissipation.

F. Discussion and Future Work

RFETs-based circuits provide unique features which can be applied for various security measures. While the inherent polymorphism can be applied to a range of security measures such as logic locking, watermarking and camouflaging techniques, the same polymorphism is also its main vulnerability and can prove to be detrimental. Probable future directions include:

- Explore how self-dual logic functions based on RFETs can contribute towards logic-locking and robustness against side-channel attacks.
- Devise measures to circumvent the security vulnerability so as to ensure circuit durability and correctness.
- Explore probable applications of such security vulnerability in terms of realizing hardware Trojans [1] or kill-switch [75].

VI. Conclusion

In the present work, we looked at various security schemes at different levels of abstraction. Firstly, we introduced ASSURE, an approach for RTL obfuscation, that allows designers to protect the semantics of hardware IP blocks at a higher level of abstraction. ASSURE is a Verilog-to-Verilog processing step that enables more semantically-meaningful protection without any changes to the existing industrial design flows.

Secondly, we discussed the requirements to evaluate the security policies at the system-level as they can detect and avoid security vulnerabilities early in the design flow. We presented static and dynamic IFT-based techniques tailored for VPs which enable the security validation of the VP as well as the embedded SW running on the VP, as demonstrated by our experiments.

Thirdly, at the circuit and architecture level, overcoming the fundamental limits of binary logic in key gates for logic locking described in MVLock can thwart foundry attacks on future-generation IPs. The MVLock technique can make it nearly impossible for an attacker to identify the design's functionality while unwarranted actors have access to the design.

Lastly, we introduced an emerging reconfigurable nanotechnology that exhibits functional polymorphism at the very transistor level. Transistors being the most fundamental piece in electronic circuits, polymorphism at this level can help in providing strong bottom-up security. We discussed how runtime-reconfigurability at the transistor-level manifests itself into interesting circuit paradigms by offering more functionality per computation unit. We demonstrate how transistor-level reconfigurability can be used for designing security primitives such as true random number generators. Finally, we introduced the underlying security vulnerability in RFETs-based circuits, which is more disruptive as it is innocuous and completely hidden in normal circuit operation. Such security vulnerability can be used to design hardware Trojans or kill-switch.

This paper aimed at giving an overview of various security practises across various abstraction levels and to introduce the readers to the state-of-the-art in IP protection. We also presented few interesting techniques which are applicable in the near future (*Quo Vadis*) to provide further security guarantees.

References

[1] S. Bhunia et al. "Hardware Trojan Attacks: Threat Analysis and Countermeasures". In: *Proc. of the IEEE* 102.8 (2014), pp. 1229–1247.

[2] J. Rajendran et al. "Nano Meets Security: Exploring Nanoelectronic Devices for Security Applications". In: *Proc. of the IEEE* 103.5 (2015).

[3] M. Rostami, F. Koushanfar, and R. Karri. "A Primer on Hardware Security: Models, Methods, and Metrics". In: *Proc. of the IEEE* (2014).

[4] M. Yasin et al. "Hardware Security and Trust: Logic Locking as a Design-for-Trust Solution". In: *The IoT Physical Layer: Design and Implementation*. Ed. by I. A. M. Elfadel and M. Ismail. Cham: Springer International Publishing, 2019, pp. 353–373.

[5] C. Pilato et al. "TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis". In: *DAC*. 2018, pp. 1–6.

[6] C. Pilato et al. "ASSURE: RTL Locking Against an Untrusted Foundry". In: *arXiv* (2020).

[7] C. Collberg, C. Thomborson, and D. Low. *A taxonomy of obfuscating transformations*. Tech. rep. 148. Department of Computer Science, The University of Auckland, New Zealand, 1997.

[8] M. E. Massad et al. "Logic Locking for Secure Outsourced Chip Fabrication: A New Attack and Provably Secure Defense Mechanism". In: *arXiv* (2017).

[9] C. Karfa et al. "Is Register Transfer Level Locking Secure?" In: *DATE*. 2020, pp. 550–555.

[10] G. E. Suh et al. "Secure Program Execution via Dynamic Information Flow Tracking". In: *ASPLOS*. 2004, pp. 85–96.

[11] D. Hedin and A. Sabelfeld. “A Perspective on Information-Flow Control”. In: *Software Safety and Security - Tools for Analysis and Verification*. 2012, pp. 319–347.

[12] M. Hassan et al. “Early SoC Security Validation by VP-based Static Information Flow Analysis”. In: *ICCAD*. 2017, pp. 400–407.

[13] P. Pieper et al. “Dynamic Information Flow Tracking for Embedded Binaries using SystemC-based Virtual Prototypes”. In: *DAC*. 2020.

[14] T. Schuster et al. “SoCRocket - A virtual platform for the European Space Agency’s SoC development”. In: *ReCoSoC*. 2014, pp. 1–7.

[15] *SoCRocket: Transaction-Level Modeling Framework for Space Applications*. https://socrocket.github.io/.

[16] V. Herdt et al. “Extensible and Configurable RISC-V based Virtual Prototype”. In: *FDL*. 2018, pp. 5–16.

[17] V. Herdt et al. “RISC-V based Virtual Prototype: An Extensible and Configurable Platform for the System-level”. In: *JSA* (2020).

[18] *RISC-V VP*. https://github.com/agra-uni-bremen/riscv-vp.

[19] J. Wilander and M. Kamkar. “A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention”. In: *NDSS*. 2003.

[20] C. Palmiero et al. “Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications”. In: *HPEC*. 2018.

[21] V. Herdt et al. “Early Concolic Testing of Embedded Binaries with Virtual Prototypes: A RISC-V Case Study”. In: *DAC*. 2019, pp. 1–6.

[22] V. Herdt et al. “Verification of Embedded Binaries using Coverage-guided Fuzzing with SystemC-based Virtual Prototypes”. In: *GLSVLSI*. 2020, pp. 101–106.

[23] S. Tempel, V. Herdt, and R. Drechsler. “An Effective Methodology for Integrating Concolic Testing with SystemC-based Virtual Prototypes”. In: *DATE*. 2021.

[24] D. Šišejković et al. “Control-Lock: Securing Processor Cores Against Software-Controlled Hardware Trojans”. In: *GLSVLSI*. 2019, 27–32.

[25] D. Šišejković et al. “A Unifying Logic Encryption Security Metric”. In: *SAMOS*. 2018, 179–186.

[26] J. A. Roy, F. Koushanfar, and I. L. Markov. “EPIC: Ending Piracy of Integrated Circuits”. In: *DATE*. 2008, pp. 1069–1074.

[27] M. Yasin and O. Sinanoglu. “Evolution of logic locking”. In: *VLSI-SoC*. 2017, pp. 1–6.

[28] D. Šišejković et al. “Inter-Lock: Logic Encryption for Processor Cores Beyond Module Boundaries”. In: *ETS*. 2019, pp. 1–6.

[29] S. Amir et al. “Comparative Analysis of Hardware Obfuscation for IP Protection”. In: *GLSVLSI*. 2017, 363–368.

[30] D. Šišejković et al. “Scaling Logic Locking Schemes to Multi-module Hardware Designs”. In: *ARCS*. 2020, pp. 138–152.

[31] D. Šišejković et al. “A Secure Hardware-Software Solution Based on RISC-V, Logic Locking and Microkernel”. In: *SCOPES*. 2020, 62–65.

[32] D. E. Nikonov and I. A. Young. “Overview of Beyond-CMOS Devices and a Uniform Methodology for Their Benchmarking”. In: *Proc. of the IEEE* 101.12 (2013), pp. 2498–2533.

[33] F. Parveen et al. “Hybrid Polymorphic Logic Gate with 5-Terminal Magnetic Domain Wall Motion Device”. In: *ISVLSI*. 2017, pp. 1–6.

[34] S. Patnaik et al. “Advancing hardware security using polymorphic and stochastic spin-hall effect devices”. In: *DATE*. 2018, pp. 97–102.

[35] Y. Gao et al. “Emerging Physical Unclonable Functions With Nanotechnology”. In: *IEEE Access* 4 (2016), pp. 61–80.

[36] H. Jiang et al. “A provable key destruction scheme based on memristive crossbar arrays”. In: *Nature Electronics* 1.10 (2018), pp. 548–554.

[37] V. C. Patil and S. Kundu. “On Leveraging Multi-threshold FinFETs for Design Obfuscation”. In: *ISVLSI*. 2020, pp. 108–113.

[38] P. Estrela, V. Pachauri, and S. Ingebrandt. “Biologically sensitive field-effect transistors: from ISFETs to NanoFETs”. In: *Essays in Biochemistry* 60.1 (June 2016), pp. 81–90.

[39] X. T. Vu et al. “Top-down processed silicon nanowire transistor arrays for biosensing”. In: *physica status solidi (a)* 206.3 (2009), pp. 426–434.

[40] S Schäfer et al. “Time-dependent observation of individual cellular binding events to field-effect transistors”. In: *Biosens. Bioelectron.* 24.5 (2009), pp. 1201–1208.

[41] A. Müller et al. “Wafer-Scale Nanoimprint Lithography Process Towards Complementary Silicon Nanowire Field-Effect Transistors for Biosensor Applications”. In: *physica status solidi (a)* 215.15 (2018).

[42] S Ingebrandt et al. “Label-free detection of single nucleotide polymorphisms utilizing the differential transfer function of field-effect transistors”. In: *Biosens. Bioelectron.* 22.12 (2007), pp. 2834–2840.

[43] K. Zamiri Azar et al. “Threats on Logic Locking: A Decade Later”. In: *GLSVLSI*. 2019, 471–476.

[44] D. Sisejkovic et al. “Challenging the Security of Logic Locking Schemes in the Era of Deep Learning: A Neuroevolutionary Approach”. In: *CoRR* abs/2011.10389 (2020). arXiv: 2011.10389.

[45] M. D. Marchi et al. “Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs”. In: *IEDM*. 2012.

[46] A. Heinzig et al. “Reconfigurable Silicon Nanowire Transistors”. In: *Nano Letters* (2012).

[47] J. Trommer et al. “Material Prospects of Reconfigurable Transistor (RFETs)–From Silicon to Germanium Nanowires”. In: *MRS Online Proceedings Library Archive* (2014).

[48] S. Tanachutiwat et al. “Reconfigurable multi-function logic based on graphene p-n junctions”. In: *DAC*. 2010, pp. 883–888.

[49] Y.-M. Lin et al. “High-performance carbon nanotube field-effect transistor with tunable polarities”. In: *IEEE TNANO* 4.5 (2005), pp. 481–489.

[50] S. Nakaharai et al. “Electrostatically reversible polarity of ambipolar α-MoTe2 transistors”. In: *ACS Nano* 9.6 (2015), pp. 5976–5983.

[51] G. V. Resta et al. “Polarity control in WSe2 double-gate transistors”. In: *Scientific reports* 6 (2016), p. 29448.

[52] S. Rai et al. “A physical synthesis flow for early technology evaluation of silicon nanowire based reconfigurable FETs”. In: *DATE*. 2018.

[53] M. Simon et al. “Top-Down Technology for Reconfigurable Nanowire FETs With Symmetric On-Currents”. In: *IEEE TNANO* (2017).

[54] T Mikolajick et al. “The RFET-a reconfigurable nanowire transistor and its application to novel electronic circuits and systems”. In: *SST* (2017).

[55] P. Ye, T. Ernst, and M. V. Khare. “The last silicon transistor: Nanosheet devices could be the final evolutionary step for Moore’s Law”. In: *IEEE Spectrum* 56.8 (2019), pp. 30–35.

[56] S. Rai et al. “Security Promises and Vulnerabilities in Emerging Reconfigurable Nanotechnology-Based Circuits”. In: *IEEE TETC* (2020).

[57] S. Rai et al. “Emerging reconfigurable nanotechnologies: Can they support future electronics?” In: *ICCAD*. 2018, pp. 1–8.

[58] Q. Alasad, J.-S. Yuan, and P. Subramanyan. “Strong Logic Obfuscation with Low Overhead against IC Reverse Engineering Attacks”. In: *ACM TODAES* 25.4 (2020), pp. 1–31.

[59] A. Chen et al. “Using emerging technologies for hardware security beyond PUFs”. In: *DATE*. 2016.

[60] S. Rai et al. “Designing efficient circuits based on runtime-reconfigurable field-effect transistors”. In: *IEEE TVLSI* 27.3 (2018), pp. 560–572.

[61] J. Trommer et al. “Functionality-Enhanced Logic Gate Design Enabled by Symmetrical Reconfigurable Silicon Nanowire Transistors”. In: *IEEE TNANO* (2015).

[62] M. Raitza et al. “Exploiting transistor-level reconfiguration to optimize combinational circuits”. In: *DATE*. 2017.

[63] S. Rai, M. Raitza, and A. Kumar. “Technology mapping flow for emerging reconfigurable silicon nanowire transistors”. In: *DATE*. 2018, pp. 767–772.

[64] S. Rai et al. “DiSCERN: Distilling Standard-Cells for Emerging Reconfigurable Nanotechnologies”. In: *DATE*. 2020, pp. 674–677.

[65] J. T. McDonald et al. “Functional polymorphism for intellectual property protection”. In: *HOST*. 2016.

[66] A. Rupani, S. Rai, and A. Kumar. “Exploiting Emerging Reconfigurable Technologies for Secure Devices”. In: *DSD*. 2019, pp. 668–671.

[67] P. Subramanyan. *Evaluating the Security of Logic Encryption Algorithms*. https://bitbucket.org/spramod/host15-logic-encryption. 2017.

[68] F. Corno, M. S. Reorda, and G. Squillero. “RT-level ITC’99 benchmarks and first ATPG results”. In: *IEEE Design & Test of computers* 17.3 (2000), pp. 44–53.

[69] P. C. Kocher. “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”. In: *CRYPTO*. 1996, pp. 104–113.

[70] P. Kocher, J. Jaffe, and B. Jun. “Differential power analysis”. In: *CRYPTO*. 1999, pp. 388–397.

[71] M. M. Sharifi et al. “A novel TIGFET-based DFF design for improved resilience to power side-channel attacks”. In: *DATE*. 2020, pp. 1–6.

[72] E. Giacomin and P.-E. Gaillardon. “Differential Power Analysis Mitigation Technique Using Three-Independent-Gate Field Effect Transistors”. In: *VLSI-SoC*. 2018, pp. 107–112.

[73] S. Rai et al. “Hardware Watermarking Using Polymorphic Inverter Designs Based On Reconfigurable Nanotechnologies”. In: *ISVLSI*. 2019.

[74] Y. Bi et al. “Emerging Technology-Based Design of Primitives for Hardware Security”. In: *J. Emerg. Technol. Comput. Syst.* (2016).

[75] S. Adee. “The hunt for the kill switch”. In: *IEEE Spectrum* 45.5 (2008), pp. 34–39.