Proof-Carrying Hardware-Based Information Flow Tracking in Analog/Mixed-Signal Designs

Mohammad Mahdi Bidmeshki, Angelos Antonopoulos, *Member, IEEE*, and Yiorgos Makris, *Senior Member, IEEE*

***Abstract*— Information flow tracking (IFT) is a widely used methodology for ensuring data confidentiality and/or integrity in electronic systems and many such methods have been developed at various software or hardware description levels. Among them, Proof-Carrying Hardware Intellectual Property (PCHIP) introduced an IFT methodology for digital hardware designs described in hardware description languages (HDLs). However, it is not only the digital domain that suffers from the risk of inadvertent information leakage. Indeed, analog signals originating from sources of sensitive information such as biometric sensors, as well as analog circuit outputs could also carry confidential information. Moreover, analog circuits are equally susceptible as their digital counterparts to malicious modifications, known as hardware Trojans, which could introduce covert channels for leaking such confidential information. Furthermore, in analog/mixed-signal circuits, such information leakage channels may cross the analog/digital or digital/analog interface, making their detection even harder and, thereby, intensifying this security concern. As a solution, we introduce a PCHIP-based methodology which enables systematic formal evaluation of information flow policies in analog/mixed-signal designs. This solution can reason on analog designs described at the transistor-level or at the block-level, where an abstract model of the analog circuit is considered. Additionally, it can handle analog circuit models developed in Verilog-A or Verilog-AMS, thereby enabling the use of circuit models developed in these HDLs for IFT purposes. By integrating IFT across the digital and analog domains, the proposed solution is able to detect sensitive data leakage from the digital domain to the analog domain and vice-versa, without requiring any modification of the current analog/mixed-signal circuit design flow.**

***Index Terms*— Information flow tracking, analog/mixed-signal design, hardware trust, hardware Trojans, information leakage.**

I. Introduction

INFORMATION flow tracking (IFT) [1] is a methodology for tracking the propagation and/or the usage of sensitive or untrusted data in computer systems. The main objective of IFT is to ensure the confidentiality and/or the integrity of sensitive data, by verifying that they do not get contaminated by untrusted sources and/or they do not reach unauthorized sites. In its basic but fundamental form, IFT augments each data element with sensitivity tags. Additionally, it defines rules (known as information flow policies) for propagating and manipulating these sensitivity tags in accordance with the operations performed on their corresponding data elements, and it restricts the usage of data with specific tags to authorized sites. Initial IFT methodologies focused on software [1] while considering hardware as the root of trust. Later efforts in this domain sought to take advantage of hardware entities to improve IFT performance [2]. Even more recently, such efforts have been further driven by the realization that hardware vulnerabilities, introduced either through inadvertent errors or through malicious tampering, can lead to major security risks [3]. As a result, several IFT-based methods [4]–[12] were introduced to evaluate and ensure the security of hardware at various abstraction levels.

Existing hardware IFT methodologies are limited to designs in the digital domain and lack support for any type of analog computation. Yet, analog and mixed-signal designs are also susceptible to hardware attacks, whereby confidentiality and/or integrity of sensitive data may be endangered. In fact, leaking secret information from the digital domain through systematic modification of analog performances has already been successfully demonstrated in previous studies [13], wherein carrier frequency or transmission power manipulation in an RF transmitter was used to leak secret encryption key data. As mixed-signal IC designs become widespread due to the ubiquitousness of wireless technologies, such as Bluetooth and Wi-Fi, and as simple digital I/Os of the past are being substituted by high-speed links which extensively combine analog and digital techniques for noise reduction or channel distortion compensation [14], this problem exacerbates. In addition, as mixed-signal designs become more complicated, adversaries are afforded more opportunities for implanting such malicious capabilities, which often require no more than a single transistor or capacitor [13], [15]. Furthermore, design errors in the analog/mixed-signal portion of a circuit, which as reported by the author in [16], account for approximately 20% of all bugs in the design cycle of modern microprocessors, may also pose security threats. Therefore, developing an IFT approach which can handle analog/mixed-signal designs

Manuscript received December 3, 2020; revised March 3, 2021; accepted April 16, 2021. Date of publication April 22, 2021; date of current version June 14, 2021. This work was supported in part by the National Science Foundation (NSF) under Grant 1318860. This article was recommended by Guest Editor J. M. Fung. *(Corresponding author: Mohammad Mahdi Bidmeshki.)*

Mohammad Mahdi Bidmeshki and Yiorgos Makris are with the Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: bidmeshki@utdallas.edu; yiorgos.makris@utdallas.edu).

Angelos Antonopoulos was with the Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080 USA. He is now with u-blox Athens S.A., 15125 Maroussi, Greece (e-mail: aantonopoulos@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2021.3075098.

Digital Object Identifier 10.1109/JETCAS.2021.3075098

becomes paramount, as it can assist in revealing potential information leakage paths and, thereby, instilling trust in such designs.

To this end, in this paper we introduce an IFT methodology which is capable of seamlessly crossing the analog/digital boundary. More specifically, by extending the previously developed Proof-Carrying Hardware Intellectual Property (PCHIP) method from the digital domain [9], [17], [18] to the transistor-level, we create a unified framework for enforcing information flow policies in digital, analog, and mixed-signal designs. Furthermore, we introduce analog IFT capabilities at a higher level of abstraction, namely the block-level. Such capabilities are particularly important for two reasons. First, they can increase accuracy, as the very fine granularity of transistor-level IFT, in conjunction with the conservativeness of the PCHIP-based method, makes it prone to false positives. Second, they can facilitate early security evaluation of designs, long before the detailed transistor-level implementation is made available. Additionally, we enhance our PCHIP-based methodology to recognize analog constructs used for modeling analog behavior in Verilog-A and Verilog-AMS. This enables designers to leverage analog circuit models, which are usually developed for design verification, for the purpose of IFT evaluation. We acknowledge that the accuracy of block-level IFT depends on the details accounted for in the block-level models. Nevertheless, transistor-level IFT can still be used for a more detailed evaluation of the design once the transistor-level implementation is available.

The rest of this paper is organized as follows. Section II further motivates the need for IFT in mixed-signal designs by exploring an example. Section III reviews related work. Section IV depicts the threat model considered in this work. Section V briefly describes the digital PCHIP-based IFT framework. Section VI introduces the proposed IFT extension for analog designs at both the transistor- and the block-level. Section VII discusses the integration of the proposed analog IFT with the digital PCHIP-based IFT framework. Section VIII demonstrates our method's ability to reveal sensitive information leakage in three analog/mixed-signal designs. Discussion and conclusions are provided in Section IX.

II. Motivational Example

Due to existence of process variations, the performance specifications for analog and mixed signal designs are not very tight and cover a range of acceptable values. This creates many opportunities in the analog domain to hide covert channels for leaking sensitive information, without compromising the design specification. Such covert channels can vary the circuit performance parameters just enough to leak the information and remain undetected in testing. As an example, Fig. 1 shows a simple ultra-wide band (UWB) transmitter which uses advanced encryption standard (AES) to encrypt the data before transmission. However, it creates a covert channel by varying the transmission power and leaks the secret key in addition to transmitting the encrypted data [13]. Other analog performance parameters, such as carrier frequency, can also be employed for this purpose. We present a few other examples in Section VIII.

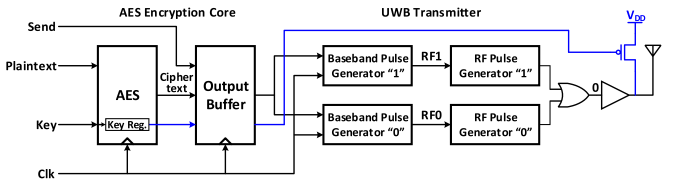


Fig. 1. A Trojan which adds a transistor in the power amplifier to leak information by varying transmission power, along with its leakage path.

Such possible exploitation of the analog domain, along with the potential existence of undetected design bugs, underline the need for a methodology that can track the flow of sensitive information in a mixed-signal design as a whole. While there are plenty of techniques applicable in the digital part, to the best of our knowledge, such a capability is lacking in the analog and mixed signal domain. This work seeks to fill this gap and is considered as a first step toward establishing a framework capable of information flow tracking in digital, analog, and mixed-signal designs.

III. Related Work

Many IFT approaches have been introduced at various software or hardware description levels. In this section, we briefly review a handful of them and contrast them to the method proposed herein.

At the software level, static IFT methods enforce information flow policies on a program at compile-time based on logical inference and reasoning [19]. In contrast, dynamic IFT schemes are applied during program execution and benefit from the availability of more detailed run-time information, yet at the cost of incurring performance and memory overhead [20]. In a different direction, towards reducing the performance overhead of IFT and improving accuracy of information flow policy enforcement, hardware-assisted IFT schemes were introduced [2]. The gate-level IFT (GLIFT) approach proposed in [4] refines the conservative rules used by higher-level IFT methods and implements more realistic tag computation operations. Thereby, it realizes a precise hardware-assisted IFT approach, at the cost of increased hardware overhead.

Various methodologies have been also introduced to evaluate trustworthiness of digital hardware designs. In this direction, several language-level approaches have been introduced, seeking to enforce information flow policies on hardware designs. For example, Caisson [5] is a hardware description language with static information flow verification capabilities at design time. Sapper [6] provides a language which employs static analysis at compile-time but also inserts dynamic checks in the resulting hardware in order to enforce information flow policies. SecVerilog [10] enforces information flow policies by introducing a type system. It is essentially Verilog, which has been extended with type annotations. SecVerilog provides an accurate information flow model by supporting dependent security types (i.e., tags defined as a function of signal values). By extending the type system of SecVerilog, authors in [11] verified isolation properties in a multi-core prototype of the ARM TrustZone architecture, demonstrating applicability to large designs with modest effort. Similarly, an extension which provides secure sharing of hardware resources at a fine granularity and does not require implicitly added hardware to enforce security, is demonstrated in [12].

A method which detects hardware Trojans leaking sensitive information based on gate-level IFT (GLIFT) [4] is introduced in [21]. At a higher level of abstraction, register transfer level IFT (RTLIFT) [7] has also been introduced to verify security properties in hardware designs. Although effective, these methodologies require exploration of the entire signal/value space using a SAT solver and/or a model checker, or incorporation in simulation based testing of the design. Along this direction, Clepsydra [8] introduced a formal modeling approach to distinguish timing-based information flows from functional flows in hardware designs described by HDL codes.

Various other approaches have been proposed in the past for formally reasoning on analog/mixed-signal functionality. For example, Booleanization of analog circuits has been utilized for formal state exploration of the design characteristics [22], [23]. Similarly, an effort to create a framework for formal verification of analog/mixed signal designs based on symbolic computation was presented in [24].

All of these methodologies, however, are either IFT solutions limited to the digital domain or are functional verification methods which do not consider IFT in the analog domain. To the best of our knowledge, the solution proposed herein is the first attempt to develop an IFT method for analog/mixed-signal circuits. Our approach creates a unified framework for IFT in analog/mixed-signal designs which can handle various levels of abstraction seamlessly, based on a previously developed PCHIP-based IFT solution for the digital domain, which we briefly review in Section V.

IV. Threat Model

The methodology described in this work seeks is to identify *sensitive information leakage* through the physical primary outputs of analog/mixed signal designs due to intentional malicious modification or inadvertent design flaws. Accordingly, we assume that the integrated design is available for analysis. Therefore, the digital parts of the design are needed as the HDL code, and the analog portions are required at the transistor-level or at higher level models. The information leakage can occur in digital or analog form. We do not consider timing side-channels in this work. Also, we do not consider, at least not directly in the proposed methodology, analog side-channels leaking to signals other than the primary outputs of a designs, e.g., side-channels created on power rails. Despite that, we do provide a mechanism for the designers to check for sensitive information leakage to internal signals of a design (except for power rails). If carrying a sensitive value was not intended for a signal, yet it is marked as sensitive by the proposed IFT approach, it could show potential existence of a side-channel on those signals. Confirmation, in this case, requires detailed mixed-signal simulations.

V. PCHIP-Based IFT in the Digital Domain

In this section, we review the fundamentals of the PCHIP-based IFT method for digital designs. The PCHIP framework [25], which is shown in Fig. 2, seeks to ensure trustworthiness of hardware designs delivered as HDL code, such as 3rd party hardware intellectual property (3PIP), by accompanying the hardware with formal proofs for a set of security properties. These properties, which are agreed upon by the developer and the consumer of the 3PIP, are crafted in a way that prevents malicious and/or unauthorized functionality in the design. In addition to preparing the hardware design, 3PIP developers are also tasked with writing proofs for these security properties, which are delivered to the consumer as part of the trusted hardware package. Since formal reasoning is not directly possible in HDLs, PCHIP defines rules for converting the hardware design to a formal representation [26], such as Coq [27], which provides an environment for mechanized proof construction and checking. In the general PCHIP framework, the exact functionality of the HDL code is converted to the formal representation. This is necessary for proving general security properties for which reasoning on the result of the computation is required, such as verifying correctness of instruction execution in a microprocessor [28].

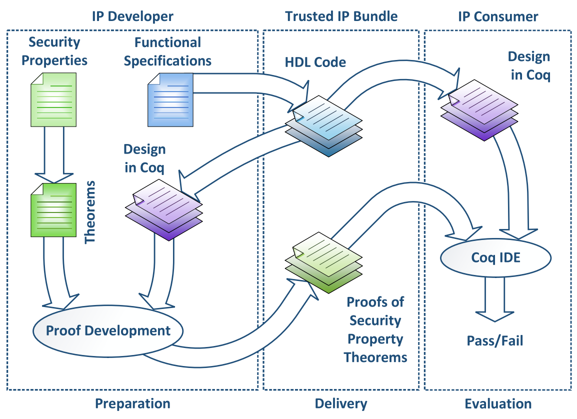


Fig. 2. PCHIP framework [25], [26].

For the purpose of enforcing information flow policies on digital hardware designs, however, the exact functionality of operations may not be necessary. Therefore, PCHIP also introduced a special framework [9], [17], [18] wherein the functionality of operators is omitted in order to simplify proof development. For example, all binary operators, such as addition, logical AND, etc., are converted to the same formal representation. Instead of focusing on the functionality, this approach maintains the exact structure of the design and focuses on the flow of information. To facilitate IFT, a sensitivity level is assigned to each signal and is maintained in a centralized sensitivity list. Information flow policies, which are defined for each operator, are then used to update the sensitivity list through evaluation of the hardware in its converted formal representation. Using this formal structural representation and the sensitivity list, security properties preventing the leakage of sensitive information in digital circuits can be developed and proven.

The process of converting the HDL design to a formal representation, developing security properties, and constructing proofs can be burdensome, as it requires significant

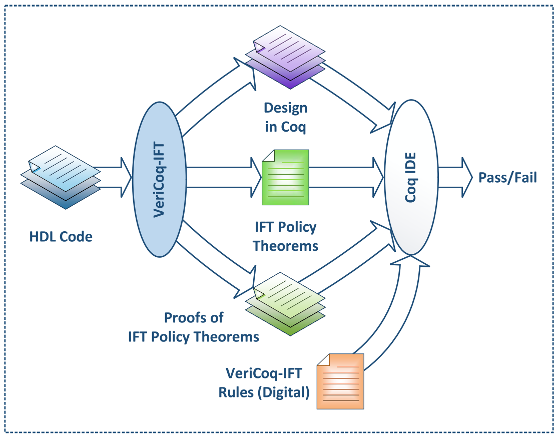


Fig. 3. Automated PCHIP framework for information flow policies [18].

effort and expertise in formal methods and proof writing. To simplify the use of PCHIP for enforcing information flow policies, we developed an automated framework named *VeriCoq-IFT* [9], [18]. As shown in Fig. 3, *VeriCoq-IFT* is able to automatically convert the Verilog design to the Coq representation, generate security property theorems for preventing sensitive information leakage and construct their proofs. It gathers all the required information, such as input sensitivity levels and sensitivity-reducing operations through special comments (pragmas) inserted into the HDL code. Therefore, designers simply need to annotate the HDL code and provide the design to *VeriCoq-IFT*. The generated proofs for the security properties are, then, automatically checked in Coq in order to verify design trustworthiness in terms of information flow policies. Rules in Fig. 3 represent functions and lemmas developed for the evaluation of the Coq representation of the design and construction of the proofs.

To elaborate further, consider the Verilog source code of Fig. 4 which defines two simple modules. The special comment in line 5 defines the initial sensitivity level of the `inp` signal as 1. A higher number means that the signal carries more sensitive information. While a procedure to determine this value is discussed in our earlier studies in [17], [18], in summary, the initial sensitivity levels are set to the minimum number of sensitivity reducing operations that a signal can go through before reaching an output in a genuine, high-level implementation of a design. The special comment in line 21 defines the eXclusive-OR operation of line 22 as a sensitivity reducer.

Fig. 5 shows the Coq representation produced by *VeriCoq-IFT* for the Verilog source code of Fig. 4. In this representation, all signals are considered as buses, identified by a natural number which shows their place in the sensitivity list. As an example, line 23 defines `inp` as number 1, occupying the second place in the sensitivity list at line 33. Note that, as expected, its initial sensitivity level is also defined as `coqSome 1`. Since there is no annotation of initial sensitivity level for other signals in the Verilog source code of Fig. 4, `coqNone` is used for their sensitivity level in the list of lines 31-36 in Fig. 5.

```
 1 module ift_sample (out, clk, inp, k);
 2   output [1:0] out;
 3   input clk;
 4
 5   /* vericoq init_sensitivity_level_1 */
 6   input [1:0] inp;
 7   input [1:0] k;
 8   reg [1:0] out;
 9   wire a, b;
10
11   always @(posedge clk)
12     out <= {a, b};
13   assign a = inp[1] & k[0];
14   my_op op1 (.out(b), .i1(inp[0]), .i2(k[1]));
15 endmodule
16
17 module my_op (out, i1, i2);
18   output out;
19   input i1, i2;
20
21   /* vericoq sensitivity_reducer */
22   assign out = i1 ^ i2;
23 endmodule
```

Fig. 4. An example Verilog source code.

```
 1 Require Import Vericoq_ift.
 2
 3 Inductive module :=
 4  | module_ift_sample : bus->bus->bus->
 5      bus->bus->bus->module->module
 6  | module_my_op : bus->bus->bus->module
 7 .
 8 Fixpoint module_inst (m:module)  :=
 9  match m with
10  | (module_ift_sample out clk inp k b a
11      module_my_op_op1)=>
12         (assign out (ecat (econb a) (econb b)));
13         (assign a (ebinop (econb (inp [1, 1]))
14           (econb (k [0, 0]))));
15       (module_inst module_my_op_op1)
16
17  | (module_my_op out i1 i2) =>
18       (assign out
19         (ereduce (ebinop (econb i1) (econb i2))))
20 end.
21
22 Definition clk : bus := Id 0.
23 Definition inp : bus := Id 1.
24 Definition k : bus := Id 2.
25 (* ... *)
26
27 Definition ift_sample:=
28  module_inst (module_ift_sample out clk inp k b a
29    (module_my_op b (inp [0, 0]) (k [1, 1]))).
30
31 Definition init_state : sen_list :=
32   ((0, None)::nil) :: (* clk *)
33   ((0, Some 1)::(0, Some 1)::nil) :: (* inp *)
34   ((0, None)::(0, None)::nil) :: (* k *)
35   (* ... *)
36   nil.
37
38 Definition check_sensitivity t :=
39   check_code_sen ift_sample init_state t.
40 Definition stable :=
41   find_stable_list ift_sample init_state 30.
42 Definition is_safe_bef_stable_out :=
43   is_safe_bef_stable out
44     ift_sample init_state (fst stable).
45
46 Theorem out_secrecy : forall (t : nat),
47   ((fst stable) < t) ->
48     is_safe_op_bus_sensitivity
49       (read out (check_sensitivity t))
50   /\ is_safe_bef_stable_out.
51 Proof.
52   intros. split.
53   assert (get_sen_val_sen_list (check_sensitivity t) =
54     get_sen_val_sen_list (check_sensitivity (fst stable))).
55   apply check_code_sen_eq_st.
56   vm_compute. reflexivity. apply H. simpl. omega.
57   assert (get_sen_val_op_bus
58     (read out (check_sensitivity t)) =
59       get_sen_val_op_bus
60         (read out (check_sensitivity (fst stable)))).
61   apply read_sen_eq_st. apply H0.
62   assert (is_safe_op_bus_sensitivity
63     (read out (check_sensitivity t)) =
64       is_safe_op_bus_sensitivity
65         (read out (check_sensitivity (fst stable)))).
66   apply op_bus_same_sen_val_is_safe. apply H1.
67   rewrite H2. vm_compute. tauto. vm_compute. tauto.
68 Qed.
```

Fig. 5. Coq representation, security property theorem and proof generated by *VeriCoq-IFT* for Verilog code of Fig. 4.

Lines 3-20 in Fig. 5 define module types and the structure of Verilog modules in Coq representation. The `ebinop` constructor is used for binary operations, such as in lines 13 and 22 in the Verilog source, while `econb` is used to convert a `bus` to an expression. The `ereduce` constructor in line 19 is used for a sensitivity reducing operation in the Coq representation.

The `check_code_sen` function used in line 39 of Fig. 5 evaluates the code in the Coq representation based on the initial sensitivity list and a conservative data flow model defined in *VeriCoq-IFT*, and returns an updated list. Since there is no sensitivity enhancing operation in this framework, evaluation of the Coq representation will eventually result in a list where further evaluations do not change the sensitivity values. *VeriCoq-IFT* calls this a stable list and uses it to prove the security property theorems. The Coq representation of Fig. 5 also shows the security property for the `out` signal in lines 46-50. This theorem ensures that the sensitivity level of this signal remains zero at all times, meaning that `out` does not carry sensitive information. The proof of this theorem is presented in lines 51-68 and is based on code evaluation (before reaching the stable list) and induction (after reaching stability). Verification of the proof for this design fails in Coq, since one bit of the sensitive input `inp` does not go through a sensitivity reducing operation before reaching `out`. We note that it is possible to generate such theorems for any signal in the design and verify its sensitivity level.

VI. IFT IN ANALOG SIGNAL DESIGNS

While PCHIP-based IFT and *VeriCoq-IFT* have been shown to be very useful and effective in digital designs, such as cryptographic hardware, they are unable to handle analog/mixed-signal circuits. To this end, in this section, we follow the conservative approach used in the digital domain [9], [17], [18] and we develop similar information flow models for handling analog signals. Together with the information flow policies of the digital domain, these models enable the application of PCHIP-based IFT to analog/mixed-signal designs. We introduce IFT for analog circuits at two levels of abstraction, namely the transistor-level IFT in Section VI-A and the block-level IFT in Section VI-B. In addition, in Section VI-C, we discuss a possible sensitivity reducing operation, which is used in the PCHIP-based IFT framework to decrease the sensitivity level of signals resulting from specific operations in the analog domain.

A. Transistor-Level IFT

Unlike digital designs which make extensive use of standard cell libraries, analog designs are commonly handcrafted at the transistor level. To enable IFT in the analog domain, we need to either perform IFT at the transistor level or model high-level analog modules (e.g. amplifiers, mixers, etc.) from the transistor level design and perform IFT at the block level (Section VI-B). In order to gain a fundamental understanding of analog signal flows at the finest granularity, in this section we consider implementing analog IFT at the transistor-level. Below, we list several considerations of transistor-level analog IFT:

- In analog circuits, information may be carried not only through voltage but also through current.
- Unlike in digital designs, wherein transistors are used as switches, analog circuits involve transistors in various configurations. For example, a MOSFET in an amplifier can be used in a common source, common gate or common drain configuration, as shown in Fig. 6, thereby producing gate-to-drain, drain-to-source, or gate-to-source data flows, respectively. Moreover, changing voltage on the source or the drain impacts the drain-to-source current. Similarly, a bipolar transistor can also be utilized in several configurations, as shown in Fig. 7, thereby creating various corresponding data flows. A transistor may also be used as an active load or a capacitor.
- The voltage on the bulk terminal of a transistor may also be manipulated to leak information to the source or drain terminals.
- Other components, such as capacitors, resistors, etc. should also be considered for information flow.

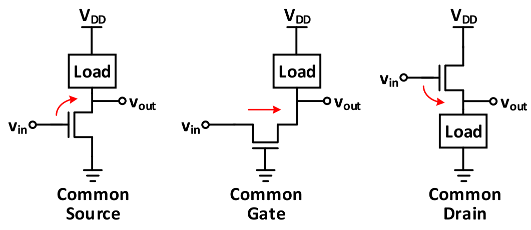


Fig. 6. MOSFET configurations in amplifiers and possible data flows.

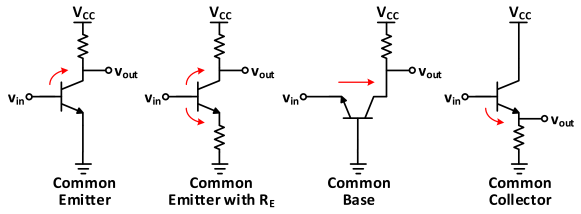


Fig. 7. BJT configurations in amplifiers and possible data flows.

Based on these considerations, we define our information flow policies for analog circuits as listed in the following:

MOSFETs: a) Any sensitive value on the gate terminal is transferred to the drain and the source. b) Any sensitive value on the bulk terminal is transferred to the drain and the source. c) Any sensitive value on the source terminal is transferred to the drain and vice versa.

Bipolar Transistors: a) Any sensitive value on the base terminal is transferred to the emitter and the collector terminals. b) Any sensitive value on the emitter terminal is transferred to the collector and vice versa.

Capacitors, inductors and resistors: Any sensitive value on one of the terminals is transferred to the other terminal of these components. The reason for this is that such components usually do not have polarity and can be used in any orientation. If required, data flow in transformers can also be defined similarly by considering them as multi-terminal components.

Diodes: Since a voltage change on any terminal of a diode can change the current through it, diodes in our conservative data flow model are treated similar to resistors and capacitors.

In any of these components, if a terminal is connected to the power supply, we ignore possible data flow to it.

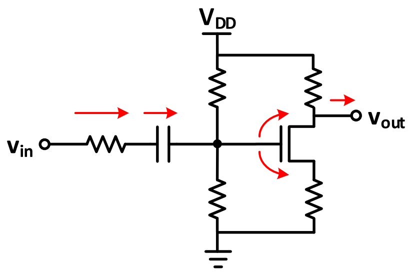


Fig. 8. Data flow from the input to the output of an amplifier through resistor, capacitor and NMOS transistor.

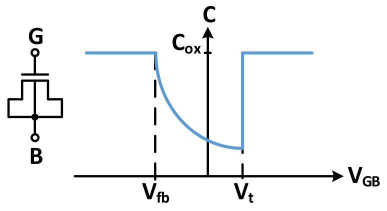


Fig. 9. MOSFET used as a capacitor at various voltages [31]. V_t and V_{fb} are threshold and flat-band voltages respectively.

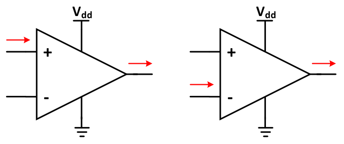


Fig. 10. Block-level information flow in a differential amplifier.

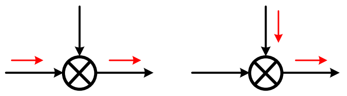


Fig. 11. Block-level information flow in an analog mixer.

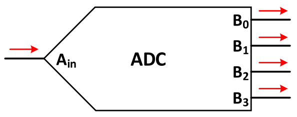


Fig. 12. Block-level information flow in an analog to digital converter (ADC).

Therefore, nodes connected to the power supply cannot carry sensitive data in our information flow model. Note that one sneaky method of leaking information is through side-channels created by systematic variation of currents in internal nodes of a circuit [29], [30]. While, in this model, our methodology cannot automatically reveal such side-channels, we provide a mechanism for manual designer annotation [18], in which suspicious internal nodes can be marked; subsequently, theorems attesting that those nodes do not leak any sensitive information can be generated and evaluated automatically in our framework.

In cases where a MOSFET is used as a capacitor, its source, drain and bulk terminals are shorted and form one terminal of the capacitor. The gate of the transistor serves as the other terminal of the capacitor, as shown in Fig. 9. Typically, in MOSFETs we do not consider a data flow to the transistor gate. However, if a transistor is used as a capacitor and none of its terminals are connected to the power supply, such as in AC coupling capacitors depicted in Fig. 8, such a flow may exist and may be missed by our model. Indeed, as shown in Fig. 9, the gate-bulk voltage does not have polarity restrictions; hence, a transistor can be used as a capacitor in any orientation. Therefore, before performing IFT, we replace all transistors whose source, drain and bulk terminals are shorted, with a capacitor.

B. Block-Level IFT

In the previous section, we introduced a transistor-level IFT approach for analog/mixed-signal designs. IFT at such low level can detect all possible flows of sensitive information. However, due to its conservative approach, it increases the chance of false positives, i.e., situations where a flow does not exist but IFT at the transistor-level detects such a flow. Also, a transistor level circuit may not be available at the early stages of the design. In addition, AMS bugs, which may also lead to security vulnerabilities, often occur at the interface between digital and analog components [16]. Therefore, it is useful to raise the level of abstraction for IFT in analog designs, in order to provide flexibility for the designers, create a means for early evaluation of the design, and reduce the chance of false positives.

Similar to transistor-level components, we can devise rules for the flow of information in analog blocks, e.g., in an amplifier or in an analog mixer. As an example, Fig. 10 shows the information flow in a differential amplifier at the block-level. As can be seen, a sensitive value on any of the inputs is transferred to the output of the amplifier. Similarly, Fig. 11 shows the information flow in an analog mixer. As another example, Fig. 12 shows the block-level information flow in a 4-bit analog-to-digital converter (ADC), in which a sensitive analog input is transferred to all output bits. Information flow in other analog blocks can also be defined accordingly.

Another way of implementing a block-level IFT for analog designs is to employ models developed in Verilog-A/AMS. To facilitate verification of large mixed-signal designs, designers may develop models representing the functionality of analog blocks using analog modeling constructs in Verilog-A/AMS. Commonly, these analog models are simulated in conjunction with the corresponding transistor-level circuit to ensure consistency of the models with the actual implementation of the circuit. These models can also be utilized for our IFT task. In the following, we describe how the fundamental Verilog-A/AMS constructs can be converted to the Coq representation for the purpose of IFT, based on our conversion methodology for digital designs.

Specifically, by treating the real and integer variables as `bus`, we can convert assignments to these variables similar to the assignments to the digital signals. However, analog contribution statements which are used inside `analog` constructs in Verilog-A/AMS require further elaboration.

In Verilog-A/AMS, a branch is defined as a path between two nets. The branch contribution operator `<+` is used to describe analog behavior [32]. The right-hand side of this operator is an expression which can be evaluated as a real value. The left-hand side specifies the signal that the right-hand side will be assigned and consists of a signal access function applied to a branch. For example:

```
V(n1, n2) <+ expression;
```

or

```
I(n1, n2) <+ expression;
```

In these examples, `(n1,n2)` represents an unnamed source branch, `V(n1, n2)` represents the potential on the branch (i.e., voltage), and `I(n1,n2)` refers to the flow through the branch (i.e., current). Note that, if the second net is omitted, the global reference node (ground) is considered as the reference net. As an example:

```
V(out) <+ Gain * V(inp);
```

sets the voltage on the `out` node (i.e., signal) to the value of `Gain` parameter times the voltage on the `inp` node. Since the second node in `V(out)` is omitted, the ground node would be the reference.

To convert these analog contribution statements to the Coq representation for IFT purposes, we break them down and convert them to two separate assignments to `n1` and `n2`. Therefore, any sensitive value on the right-hand side will be transferred to both nets. Note that all nodes and signals, discrete or continuous, are considered as buses in our IFT approach. Also, for function calls, such as sin, ddt, etc., as well as signal access functions used on the right-hand side, we consider the sensitivity level of the function parameters to evaluate the resulting sensitivity of the expression. In addition, indirect branch assignments, such as

```
V(out): V(in) == 0;
```

which means “find `V(out)` so that `V(in) == 0`” [32], are handled similar to a regular analog contribution statement.

Based on these basic conversion rules, along with the methodology for the digital designs described in Section V, the Verilog-A/AMS model of an analog design can be converted to a Coq representation and evaluated for IFT policies. Details regarding the integration with the IFT in the digital domain are provided in Section VII.

We want to emphasize that, although IFT at the block-level reduces the complexity as compared to the transistor-level, it may miss some of the flows due to possible inaccuracy of the block-level models and, thereby, introduce false negatives. Therefore, it is advantageous to perform IFT at the block-level in early stages of the design process and recheck the transistor-level implementation when the design is finalized.

C. Sensitivity Reducing Operations in Analog

As we mentioned in Section V, *VeriCoq-IFT* defines special sensitivity reducing operations which a user needs to mark in the Verilog code. For the digital domain, we consider the eXclusive-OR operation between intermediate results and the key (or sub-keys) as a sensitivity reducer. Although encryption in the digital domain is much more secure than in the analog domain, there also exist efforts in the analog domain which may qualify some operations as sensitivity reducers.

For example, the methods suggested by the authors in [33], [34] for encryption in the analog domain consist of generating a signal with secret characteristics, such as frequency and phase. These characteristics can also be selected based on a digital code [34]. We can consider this signal as equivalent to the secret key for encryption in the digital domain. Consequently, the analog encryption is performed by combining this signal with the signal we want to encrypt, i.e., the plaintext, which is also analog, using an analog mixer, and the resulting encrypted signal is transmitted through a carrier on a public channel. Therefore, mixers in such configurations can be considered as analog sensitivity reducers.

VII. Integration With Digital PCHIP-Based IFT

To enable IFT in analog/mixed signal designs, we need an integrated IFT framework capable of handling both analog and digital designs. *VeriCoq-IFT*, which we reviewed in Section V, provides an automated PCHIP-based IFT framework for the digital domain. It receives the design as Verilog code and generates a formal representation of the design along with the security theorems needed to enforce information flow policies and their proofs, which can then be evaluated in Coq. To take advantage of this framework for our purpose, we need to: (i) have a Verilog netlist of the analog/mixed-signal design, (ii) define transistor-level/block-level analog data flow policies in *VeriCoq-IFT*, and (iii) be able to handle and use Verilog-A/AMS models for IFT.

Generating the Verilog netlist of a design is straightforward using current EDA tools for analog/mixed-signal design development. For integrated analog/mixed-signal PCHIP-based IFT, we prefer to have the digital part of the design at the register transfer-level or the gate-level, and the analog part at the transistor-level or the block-level. Conveniently, current EDA tools support designs described at various abstraction levels and provide capabilities to select the level of netlisting.

To enhance *VeriCoq-IFT* with analog data flow policies, such as the ones described in Section VI, we create modules which mimic such data flow. These modules are defined just for the purpose of IFT and do not have meaningful interpretations in the digital domain. However, by using them, *VeriCoq-IFT* is able to seamlessly handle IFT in analog/mixed-signal designs. To elaborate further, Fig. 13 shows sample module definitions for modeling IFT in capacitors, NMOS and NPN transistors. It also shows a module mimicking block-level information flow in differential amplifiers. As a simple example, consider the `nch` module which represents an NMOS transistor wherein the analog data flow has been modeled by defining two assignments for the drain and source terminals. Instead of the logical AND, any other binary operation could also be used in these assign statements.

In this approach, nodes connected to the power supply are also handled correctly based on the defined analog information flow policy; indeed, when extracting the design netlist, such nodes are constantly connected to zero or one values, hence *VeriCoq-IFT* always assigns a sensitivity level of zero to them.

```
// Modeling analog data flow in capacitors
// Resistors, inductors and diodes are defined similarly
module cap (a, b);
  inout a, b;

  assign a = a & b;
  assign b = a & b;
endmodule

// Modeling analog data flow in NMOS transistors
// PMOS transistors are defined similarly
module nch (d, b, g, s);
  input g;
  inout d, b, s;

  assign d = d & b & g & s;
  assign s = d & b & g & s;
endmodule

// Modeling analog data flow in NPN transistors
// PNP transistors are defined similarly
module npn (b, c, e);
  input b;
  inout c, e;

  assign c = b & c & e;
  assign e = b & c & e;
endmodule

// Modeling analog data flow in differential amplifiers
// at the block-level
module diff_amp (p, n, out);
  input p, n;
  output out;

  assign out = p & n;
endmodule
```

Fig. 13. Sample module definitions to mimic analog data flows in *VeriCoq-IFT*.

```
module amp(out, inp);
  inout out;
  inout inp;

  electrical out, inp;
  parameter real Gain = 1,
                 Rin = 1,
                 Cin = 1,
                 Rout = 1,
                 Lout = 1;

  analog begin
    // gain of amplifier
    V(out) <+ Gain * V(inp);

    // model input admittance
    I(inp) <+ V(inp) / Rin;
    I(inp) <+ Cin * ddt(V(inp));

    // model output impedance
    V(out) <+ Rout * I(out);
    V(out) <+ Lout * ddt(I(out));
  end
endmodule
```

Fig. 14. A simple amplifier model with parasitics in Verilog-AMS [32].

```
Inductive module :=
 | module_amp : bus->bus->nat->nat->nat->nat->nat->module
.

Fixpoint module_inst (m:module)  :=
 match m with
 | (module_amp out inp Gain Rin Cin Rout Lout) =>
    (assign out (ebinop (econb (Const None)) (econb inp)));
    (assign inp (ebinop (econb inp) (econb (Const None))));
    (assign inp (ebinop (econb (Const None)) (econb inp)));
    (assign out (ebinop (econb (Const None)) (econb out)));
    (assign out (ebinop (econb (Const None)) (econb out)))
end.
```

Fig. 15. Coq representation generated by *VeriCoq-IFT* for Verilog-AMS code of Fig. 14.

The rest of the procedure for enforcing information flow policies on a design remains the same as in the digital domain. Designers need to specify the sensitivity level of input signals and mark the sensitivity reducing operations in the design through special comments defined by *VeriCoq-IFT*. If any sensitivity reducing operation is required in the analog domain, it is also defined by annotating the corresponding modules which reflect the analog information flow. Once the annotated Verilog code of the analog/mixed-signal design is provided to *VeriCoq-IFT*, the corresponding formal representation, theorems and proofs to be verified in Coq are seamlessly generated.

To take advantage of analog models developed in Verilog-A/AMS, we enhanced *Vericoq-IFT* to be able to handle analog constructs as described in Section VI-B. As an example, Fig. 14 shows a simple amplifier model in Verilog-A/AMS. It uses several analog contribution statements to model input and output parasitics. Fig. 15 shows the corresponding Coq representation for this amplifier model generated by the analog-enhanced *VeriCoq-IFT*. As can be seen, analog contribution statements are converted to the Coq representation similar to the digital assignment statements. While parameters were added in the module definition in the Coq representation, since they represent constant values, a constant sensitivity is considered for them in the expressions.

It is worth noting that, with the enhanced *VeriCoq-IFT*, it is also possible to employ the analog models of transistor-level components such as MOSFETs or capacitors, which may have already been available in Verilog-A/AMS model libraries, as transistor-level IFT models. With this approach, we do not need to utilize modules mimicking the information flow at the transistor-level, such as the ones shown in Fig. 13.

Fig. 16 shows *VeriCoq-IFT* enhanced with IFT in the analog domain. With the enhancements, *VeriCoq-IFT* can handle analog/mixed signal designs seamlessly and facilitates IFT at various abstraction levels with minimal user intervention.

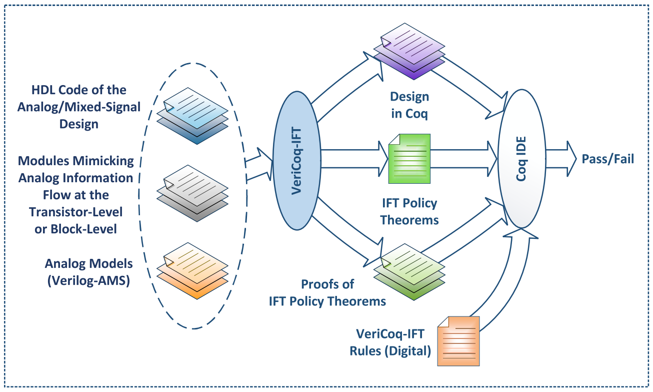

HDL Code of the Analog/Mixed-Signal Design
Modules Mimicking Analog Information Flow at the Transistor-Level or Block-Level
Analog Models (Verilog-AMS)
VeriCoq-IFT
Design in Coq
IFT Policy Theorems
Proofs of IFT Policy Theorems
VeriCoq-IFT Rules (Digital)
Coq IDE
Pass/Fail

Fig. 16. *VeriCoq-IFT* enhanced with IFT in the analog domain.

VIII. Demonstrations

In this section, we apply the proposed methodology and we demonstrate its ability to reveal information leakage paths in analog/mixed signal designs.[1] We verify the effectiveness of IFT using both transistor-level implementations and analog models developed in Verilog-AMS.

A. Information Leakage From Digital to Analog Domain

The first design that we experiment with is a wireless cryptographic IC consisting of an advanced encryption standard (AES) core and a UWB transmitter [13], whose block

[1]We clarify that this method is only able to detect such paths if they are present in any abstraction level of the design netlist. Malicious capabilities introduced after the design is sent for fabrication (i.e., via mask modification) cannot be detected unless a chip is reverse-engineered to a netlist first.

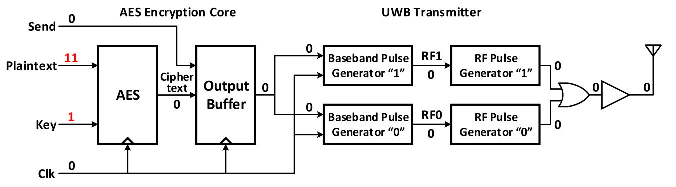


Fig. 17. The block diagram of a wireless cryptographic IC design [13]. Numbers represent the propagated sensitivity levels.

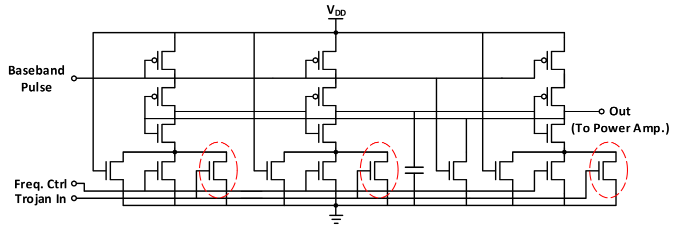
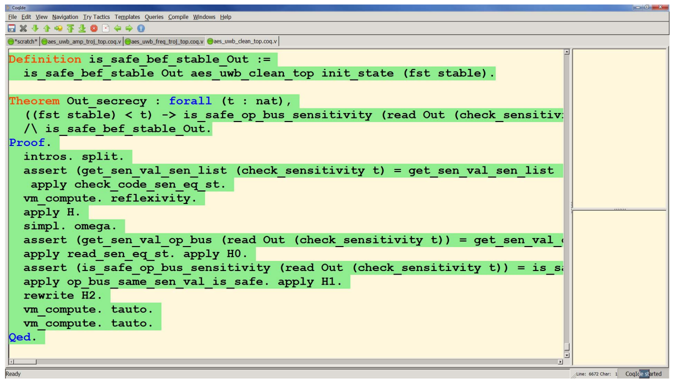


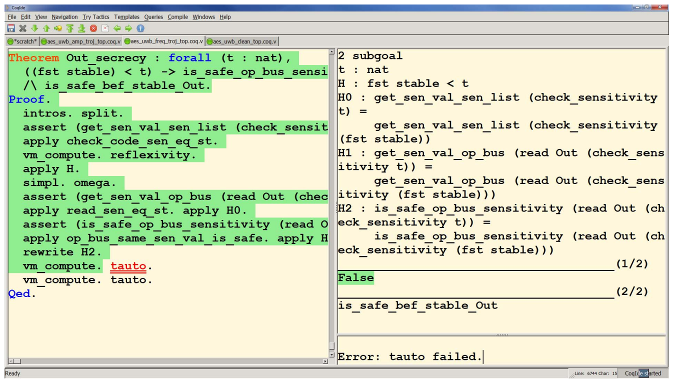
Fig. 19. Transistors added to enable information leakage through varying carrier frequencies in RF pulse generators.


CoqIde
File Edit View Navigation Try Tactics Templates Queries Compile Windows Help
scratch
aes_uwb_amp_troj_top.coq.v
aes_uwb_freq_troj_top.coq.v
aes_uwb_clean_top.coq.v
Ready

```
Definition is_safe_bef_stable_Out :=
  is_safe_bef_stable Out aes_uwb_clean_top init_state (fst stable).

Theorem Out_secrecy : forall (t : nat),
  ((fst stable) < t) -> is_safe_op_bus_sensitivity (read Out (check_sensitiv
  /\ is_safe_bef_stable_Out.
Proof.
  intros. split.
  assert (get_sen_val_sen_list (check_sensitivity t) = get_sen_val_sen_list
   apply check_code_sen_eq_st.
  vm_compute. reflexivity.
  apply H.
  simpl. omega.
  assert (get_sen_val_op_bus (read Out (check_sensitivity t)) = get_sen_val_
  apply read_sen_eq_st. apply H0.
  assert (is_safe_op_bus_sensitivity (read Out (check_sensitivity t)) = is_s
  apply op_bus_same_sen_val_is_safe. apply H1.
  rewrite H2.
  vm_compute. tauto.
  vm_compute. tauto.
Qed.
```

Fig. 18. Proof verification for the output in the clean design of Fig. 17 in Coq IDE.


CoqIde
File Edit View Navigation Try Tactics Templates Queries Compile Windows Help
scratch
aes_uwb_amp_troj_top.coq.v
aes_uwb_freq_troj_top.coq.v
aes_uwb_clean_top.coq.v
Ready
CoqIde started

```
Theorem Out_secrecy : forall (t : nat),
  ((fst stable) < t) -> is_safe_op_bus_sensi
  /\ is_safe_bef_stable_Out.
Proof.
  intros. split.
  assert (get_sen_val_sen_list (check_sensit
  apply check_code_sen_eq_st.
  vm_compute. reflexivity.
  apply H.
  simpl. omega.
  assert (get_sen_val_op_bus (read Out (chec
  apply read_sen_eq_st. apply H0.
  assert (is_safe_op_bus_sensitivity (read O
  apply op_bus_same_sen_val_is_safe. apply H
  rewrite H2.
  vm_compute. tauto.
  vm_compute. tauto.
Qed.
```

```
2 subgoal
t : nat
H : fst stable < t
H0 : get_sen_val_sen_list (check_sensitivity
t) =
     get_sen_val_sen_list (check_sensitivity
(fst stable))
H1 : get_sen_val_op_bus (read Out (check_sens
itivity t)) =
     get_sen_val_op_bus (read Out (check_sens
itivity (fst stable)))
H2 : is_safe_op_bus_sensitivity (read Out (ch
eck_sensitivity t)) =
     is_safe_op_bus_sensitivity (read Out (ch
eck_sensitivity (fst stable)))
______________________________________(1/2)
False
______________________________________(2/2)
is_safe_bef_stable_Out

Error: tauto failed.
```

Fig. 20. Proof verification fails in Coq IDE for the output in the design leaking the secret key by varying the carrier frequency.

diagram is shown in Fig. 17. Authors in [13] introduced two Trojans in this design in order to leak the secret AES key while transmitting the ciphertext (both of which are 128 bits long). This is achieved by a slight yet systematic modification of the carrier frequency or transmission power, without violating the design specifications, as we previously mentioned in Section II. Using this circuit, we demonstrate a scenario wherein sensitive information is leaked through a side-channel from the digital domain to the analog domain. A Trojan-free and two Trojan-infested versions of this design were evaluated, as we describe below.

1) *Trojan-Free Design:* This design does not contain any sensitive information leakage paths to a digital or analog output. Given the design, we first extracted its Verilog netlist and annotated the `Plaintext` and `Key` signals with appropriate sensitivity levels. We also marked the corresponding sensitivity reducing operations in the AES core. Then, using *VeriCoq-IFT*, we converted the design to the formal representation and used Coq to evaluate the automatically generated proof for the security theorem asserting the sensitivity of the design output. The proof of the security property theorem for the output passes in Coq, as shown in Fig. 18, attesting that this output never leaks sensitive information, under the provision that the initial sensitivity values and sensitivity reducing operations were annotated correctly in the design. For illustration purposes, Fig. 17 also shows the sensitivity levels propagated in this design, where '0' signifies a non-sensitive signal.

To demonstrate IFT at the block-level, we also utilized a model of the analog blocks in Verilog-AMS. Considering these models along with the digital part, we converted the design to the Coq representation utilizing *VeriCoq-IFT*. Again, checking of the proof for the output in this design passes in Coq, showing that no sensitive information is leaked.

2) *Carrier Frequency Trojan:* This design contains a hardware Trojan which uses a few added transistors in the "RF pulse generators" to vary the carrier frequency based on the value of the leaked AES key bit, which is tapped from the key storage register, as shown in Fig. 19. By evaluating this design using the enhanced *VeriCoq-IFT*, we observe that the proof does not pass in Coq, as shown in Fig. 20. Here, the `tauto` (tautology) tactic expects a `True` proposition but encounters a `False` proposition, preventing the proof checking to continue. As we specified in Section V, the theorem and its proof is devised in two parts. The first part of the theorem, which is shown below, deals with the sensitivity of the `Out` signal after reaching a stable sensitivity list, while the second part (after $\bigwedge$) considers the sensitivity of this signal before reaching a stable sensitivity list:

```
Theorem Out_secrecy: forall (t: nat),
  (fst stable) < t ->
    is_safe_op_bus_sensitivity
      (read Out (check_sensitivity t))
⋀ is_safe_bef_stable_Out.
```

In this theorem, `stable` is a tuple returned by the `find_stable_list` function and its first member, returned by `(fst stable)`, represents the time stamp at which the code evaluation reaches the stable sensitivity list. Therefore, `(fst stable) < t` indicates a precondition showing `t` is greater than the time stamp we reach the stable sensitivity list. The part after `->` enforces that the `Out` signal does not carry sensitive information, i.e., its sensitivity is zero, at all times `t`, after reaching the stable sensitivity list. After applying the

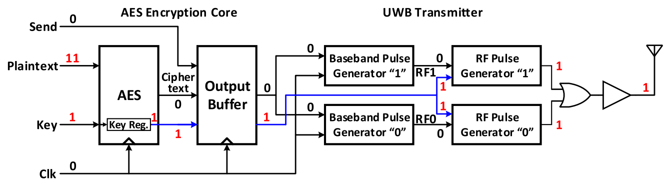


Fig. 21. Information leakage path in Trojan varying carrier frequency.

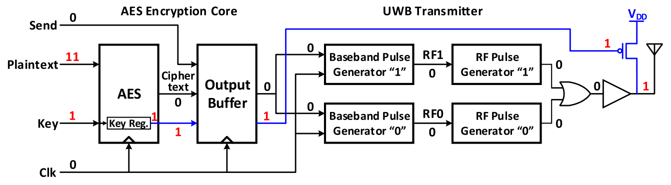


Fig. 22. Trojan of Fig. 1 with the propagated sensitivity levels.

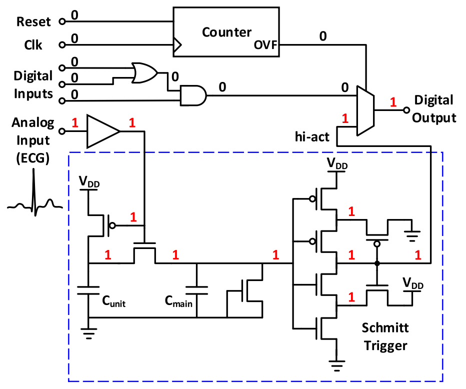


Fig. 23. An example circuit leaking information from the analog to the digital domain. The dashed box shows *A2*, which was introduced by the authors in [15]. Numbers represent the propagated sensitivity levels.

lemmas developed in *VeriCoq-IFT* to prove this part, we get to the point where we need to verify that the sensitivity of the `Out` signal is actually safe in the stable sensitivity list we found for this design. The `vm_compute` tactic tries to do this by reading the sensitivity value of the `Out` signal from the stable sensitivity list for this design and comparing it to zero. Since the stable sensitivity list we get by evaluation of this design contains a non-zero value for the `Out` signal, this tactic results in a `False` proposition, leading to a failure in the proof verification. Therefore, Coq stops the proof verification and the proof verification for the second part of the theorem is not performed.

Failure in the proof verification implies that a possible path exists through which sensitive information may leak to the output. For further insight, Fig. 21 shows the information leakage path and the propagated signal sensitivity levels in this design, where the output has a non-zero value.

Since the Trojan in this design modifies the transistor-level implementation of analog blocks, using the block-level IFT may not detect the possible information leakage unless we also model the leaking mechanism in the Verilog-AMS description. Therefore, to demonstrate the applicability of block-level IFT using *VeriCoq-IFT*, we modified the block-level models of the RF pulse generators described in Verilog-AMS and added another input which modifies the output frequency according to the transistor-level implementation. Using these models and converting the design to Coq through *VeriCoq-IFT*, we observe that the checking of the proof for the output does not pass, which again implies possible information leakage.

3) Transmission Power Trojan: As described in Section II, this design contains a hardware Trojan which adds an additional transistor at the output of the power amplifier in order to slightly increase the transmission power when the value of the leaked AES key bit is zero. Following the same procedure as in Section VIII-A.1, we evaluated this design and obtained a proof for the security property of the output, which does not pass in Coq. Once again, this implies that the proposed method detects the fact that there exists a potential for sensitive information leakage in this design. Fig. 22 shows the signal sensitivity levels in this design in addition to the information leakage path.

We also repeated the experiment utilizing the Verilog-AMS models of analog blocks. In this case, we have the design at both block-level and transistor level, since the leaking mechanism employs a single transistor which is separate from our analog block models. Therefore, we have the design as a combination of various abstraction levels: (i) transistor-level, (ii) analog block-level, and (iii) digital gate-level and register transfer-level. Again, checking the proof for the output does not pass in Coq, showing possible information leakage. This shows that our integrated methodology is capable of handling the design at various levels of abstraction.

B. Information Leakage From Analog to Digital Domain

Sensitive information may also originate from the analog domain, such as signals coming from a sensor. Therefore, in this section we demonstrate a case where sensitive information can leak from the analog domain to the digital domain, as well as the ability of the proposed approach to detect it.

Fig. 23 shows a circuit which we created to demonstrate this concept. Let us assume that the analog input in this design originates from a sensitive analog biometric source, such as an electrocardiogram signal. Peaks in this signal reflect the heart-beat and therefore reveal the activity level or the excitement level of a person. Evidently, such biomedical data is considered confidential and its inadvertent disclosure can raise privacy concerns [35]. Our example circuit borrows a technique called *A2*, which was introduced by the authors in [15] to devise an analog counter that generates a trigger signal based on the electrical activity on a victim wire. Once the on-off frequency on the victim wire reaches a certain threshold, defined by the size of the capacitors and the transistors, the output of this circuit, which is shown by a dashed box in Fig. 23, is activated. The Schmitt trigger in the design adds hysteresis to the trigger threshold in order to avoid output oscillation. Instead of using this technique as a Trojan trigger as employed by the authors in [15], we utilized it to detect a certain level of activity on the input coming from the electrocardiogram signal and digitize it on the "hi-act" signal.

The digital output in this design is not supposed to convey any information about the heart rate. However, due to this

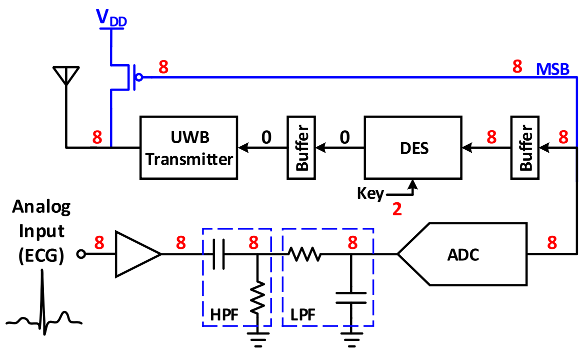


Fig. 24. A circuit leaking the MSB from the output of ADC by manipulating the transmission power.

stealthy circuit, and after a certain amount of time defined by the digital counter has lapsed, the digital output switches to signal "hi-act", thereby passing sensitive analog information to the digital domain. When embedded in a large design, this illegal behavior of the digital output may not be revealed through functional testing.

To evaluate this design using our approach, we followed a similar procedure as in Section VIII-A.1 and converted the design to its corresponding Coq representation, while marking the analog input as sensitive. Verification of the proof of the security property for the digital output in this design fails in Coq, showing that this type of information leakage is also successfully revealed by our IFT methodology. To provide more insight, numbers in Fig. 23 show the propagated sensitivity levels which lead to the digital output being marked as sensitive.

In addition to IFT at the transistor-level, we also performed block-level IFT by modeling the Schmitt trigger in Verilog-AMS and converting the design to Coq using *VeriCoq-IFT*. Similar to the transistor-level, the verification of the proof for the output in this case, which is a combination of block-level, transistor-level, gate-level and register transfer-level abstractions, does not pass in Coq, indicating possible information leakage.

As another demonstration of information flow from analog to digital, Fig. 24 shows a circuit which is supposed to send encrypted values coming from the analog input and converted to digital using an analog to digital converter (ADC). The encryption block in this design uses the Data Encryption Standard (DES) algorithm. However, through the addition of one transistor, this design leaks the most significant bit (MSB) of the ADC output using a similar leaking mechanism as the one described in Section VIII-A.3. Assuming that the analog input comes from a sensitive source such as ECG, this leaked bit can reveal private and confidential information about the user.

To evaluate this circuit using our methodology, we utilized a Verilog-AMS model for the ADC and implemented the other analog parts at the transistor level. Together with the digital blocks, we developed two implementations of this design: i) Trojan-free, and ii) Trojan infested. By evaluating the proof of the security property for the output of these designs in Coq, we observe that the proof for the Trojan-free design passes while the checking fails for the Trojan-infested design. For illustration purposes, numbers in Fig. 24 show the sensitivity values as they propagate through the Trojan-infested design.

IX. Discussion and Conclusion

We introduced an integrated methodology for IFT in analog/mixed-signal designs, which is capable of transcending various levels of design abstraction, from the register transfer level down to the transistor level. To the best of our knowledge, this is the first approach supporting IFT in analog designs. Integrated with an existing IFT method in the digital domain, our solution can help reveal possible information leakage paths in complex mixed-signal designs, which may exist either due to design errors or due to malicious modifications by an in-house adversary or an untrusted third-party IP provider. Verification of the proofs for the most complex designs in our experiments which include AES cores, on a Windows 7 computer with 8 GB of RAM and an Intel Core i7 processor, completed in about 30 minutes.

Similar to most other IFT approaches, our methodology relies on accurate labeling of sensitive signals and sensitivity reducing operations in the design. To this end, a methodology for computing the initial sensitivity levels of the signals by considering a high-level trusted implementation of the design, as well as the various considerations for annotating sensitivity reducing operations, were discussed for the digital domain in our earlier studies [17], [18]. A similar approach can also be adopted for setting the initial sensitivities in analog/mixed-signal designs. While encryption and sensitivity reduction is not as pervasive in the analog domain, it is still possible. Such an example of a common analog encryption approach proposed in the literature was discussed in Section VI-C, along with a demonstration that our methodology can handle such operations both in the digital and in the analog domain in Section VII.

A limitation of our approach, as mentioned in Section VI-A, is that it cannot automatically reveal side-channels created by varying continuous entities (e.g., currents or voltages) in internal nodes of the circuit. However, our enhanced *VeriCoq-IFT* solution enables designers to annotate suspicious internal signals and, thereby, automatically generate theorems to evaluate their sensitivity levels. As we described in Section IV, unexpected propagated sensitive labels on such internal signals can show potential existence of a side-channel, and requires further inspection and/or simulation. While this solution is not completely automated (i.e., it requires the help of annotations by the expert designer), it does provide the means for evaluating side-channels instigated by internal nodes.

Parasitic components can also create a path for information leakage, as demonstrated by the authors in [36] for the case of capacitive crosstalk. While we did not consider parasitic components in our analysis, it is possible to extract such parasitic components from the layout and include them in the design in a preprocessing step based on a minimum threshold value defined by the designer or some other criteria. However, proper analysis of parasitic components and understanding of their contribution to possible information leakage requires detailed simulations.

Lastly, *VeriCoq-IFT* and the PCHIP-based IFT methodology which we introduced herein for analog/mixed-signal designs, adopt a conservative information flow policy, which may lead to false positives although we did not encounter such false positives in our practical experiments. For example, a transistor in the off-state may block the information flow, but our current method only considers the worst possible case. As another example, even in the presence of an information flow, perturbations of an output due to changes in the sensitive signal may be so minuscule, e.g., due to sizing or component values, that they may not be realistically detectable; hence, they can be considered a false positive in our methodology. Once a flow is identified by our approach, mixed-signal simulations can be employed to filter out possible false positives, requiring manual effort by the designers. While alternative and more accurate approaches, such as GLIFT [21], RTLIFT [7], SecVerilog [10] and its descendants [11], [12] were recently introduced in the digital domain, they are currently restricted to the digital domain. We envision the future development of better IFT models for analog parts of a mixed-signal design, which will allow integration with these approaches to improve IFT accuracy in such designs. Pure digital models, e.g., using Booleanization [16], can lead to improved formal analysis-based IFT methods. On the other hand, simulation IFT models developed in Verilog-A/AMS for analog designs can enable integration with methods such as GLIFT and RTLIFT and can enable mixed-signal simulations for IFT purposes. This can lead to methodologies that take circuit functionality, component values, as well as signal values and amplitudes into account for IFT. We want to emphasize that the analog domain provides the attackers a diverse means of leaking information which might not have been utilized or identified before. Therefore, at least some degree of conservatism in defining analog information flow policies might be beneficial.

Based on the findings of this initial study, it is evident that further research is needed towards developing more accurate information flow policies, commensurate with the unique characteristics of analog/mixed signal designs. Additionally, our future efforts will be directed towards developing solutions that capture side-channels which cannot be currently automatically detected by our framework. Finally, we intent to extend the set of Verilog/Verilog-A/AMS constructs that can be handled by *VeriCoq-IFT*, in order to expand its domain of applicability. Overall, we anticipate that the robustness and effectiveness of this initial IFT solution in the analog/mixed-signal domain will inspire the hardware security community to further develop this nascent but very important area.

References

[1] A. C. Myers and B. Liskov, "A decentralized model for information flow control," in *Proc. 16th ACM Symp. Operating Syst. Princ. (SOSP)*, 1997, pp. 129–142.

[2] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas, "Secure program execution via dynamic information flow tracking," in *Proc. 11th Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS-XI)*, Oct. 2004, pp. 85–96.

[3] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, Dec. 2016.

[4] M. Tiwari, H. M. G. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood, "Complete information flow tracking from the gates up," in *Proc. 14th Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS)*, Mar. 2009, pp. 109–120.

[5] X. Li *et al.*, "Caisson: A hardware description language for secure information flow," in *Proc. ACM Conf. Program. Lang. Design Implement. (PLDI)*, 2011, pp. 109–120.

[6] X. Li *et al.*, "Sapper: A language for hardware-level security policy enforcement," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Feb. 2014, pp. 97–112.

[7] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1691–1696.

[8] A. Ardeshiricham, W. Hu, and R. Kastner, "Clepsydra: Modeling timing flows in hardware designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 147–154.

[9] M.-M. Bidmeshki and Y. Makris, "Toward automatic proof generation for information flow policies in third-party hardware IP," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 163–168.

[10] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A hardware design language for timing-sensitive information-flow security," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Mar. 2015, pp. 503–516.

[11] A. Ferraiuolo, R. Xu, D. Zhang, A. C. Myers, and G. E. Suh, "Verification of a practical hardware security architecture through static information flow analysis," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Operating Syst.*, Apr. 2017, pp. 555–568.

[12] A. Ferraiuolo, W. Hua, A. C. Myers, and G. E. Suh, "Secure information flow verification with mutable dependent types," in *Proc. 54th Annu. Design Automat. Conf.*, Jun. 2017, pp. 1–6.

[13] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1506–1519, Apr. 2017.

[14] J. Han, Y. Lu, N. Sutardja, K. Jung, and E. Alon, "Design techniques for a 60 Gb/s 173 mW wireline receiver frontend in 65 nm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 871–880, Apr. 2016.

[15] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 18–37.

[16] K. V. Aadithya, "Accurate booleanization of continuous dynamics for analog/mixed-signal design," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, 2016.

[17] Y. Jin, X. Guo, R. G. Dutta, M.-M. Bidmeshki, and Y. Makris, "Data secrecy protection through information flow tracking in proof-carrying hardware IP—Part I: Framework fundamentals," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2416–2429, Oct. 2017.

[18] M.-M. Bidmeshki, X. Guo, R. G. Dutta, Y. Jin, and Y. Makris, "Data secrecy protection through information flow tracking in proof-carrying hardware IP—Part II: Framework automation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2430–2443, Oct. 2017.

[19] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 1, pp. 5–19, Jan. 2003.

[20] L. Lam and T.-C. Chiueh, "A general dynamic information flow tracking framework for security applications," in *Proc. 22nd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 2006, pp. 463–472.

[21] W. Hu, B. Mao, J. Oberg, and R. Kastner, "Detecting hardware trojans with gate-level information-flow tracking," *Computer*, vol. 49, no. 8, pp. 44–52, Aug. 2016.

[22] A. V. Karthik and J. Roychowdhury, "ABCD-L: Approximating continuous linear systems using Boolean models," in *Proc. 50th Annu. Design Automat. Conf. (DAC)*, May 2013, pp. 1–9.

[23] A. V. Karthik, S. Ray, P. Nuzzo, A. Mishchenko, R. Brayton, and J. Roychowdhury, "ABCD-NL: Approximating continuous non-linear dynamical systems using purely Boolean models for analog/mixed-signal verification," in *Proc. 19th Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2014, pp. 250–255.

[24] M. H. Zaki, O. Hasan, S. Tahar, and G. Al-Sammane, "Framework for formally verifying analog and mixed-signal designs," in *Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design*. Cham, Switzerland: Springer, 2015, pp. 115–145.
[25] E. Love, Y. Jin, and Y. Makris, "Proof-carrying hardware intellectual property: A pathway to trusted module acquisition," *IEEE Trans. Inf. Forensics Security.*, vol. 7, no. 1, pp. 25–40, Feb. 2012.
[26] M.-M. Bidmeshki and Y. Makris, "VeriCoq: A Verilog-to-Coq converter for proof-carrying hardware automation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 29–32.
[27] INRIA. *The Coq Proof Assistant*. Accessed: Nov. 10, 2020. [Online]. Available: https://coq.inria.fr/
[28] Y. Jin and Y. Makris, "A proof-carrying based framework for trusted microprocessor IP," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 824–829.
[29] Trust-Hub. *Trust-HUB: An Online Community for Hardware Security and Trust*. Accessed: Nov. 15, 2020. [Online]. Available: https://www.trust-hub.org/
[30] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2009, pp. 117–122.
[31] C. Hu, *Modern Semiconductor Devices for Integrated Circuits*. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
[32] Accellera. *Verilog-AMS Language Reference Manual*. Accessed: Nov. 10, 2020. [Online]. Available: https://accellera.org/images/downloads/standards/v-ams/VAMS-LRM-2-4.pdf
[33] J. Seitner, "Analog scrambler," U.S. Patent 6 973 188, Dec. 6, 2005. [Online]. Available: https://www.google.com/patents/US6973188
[34] H. Babb and M. Brooks, "Analog encryption," U.S. Patent 7 545 929, Jun. 9, 2009. [Online]. Available: https://www.google.com/patents/US7545929
[35] A. Raij, A. Ghosh, S. Kumar, and M. Srivastava, "Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment," in *Proc. Annu. Conf. Hum. Factors Comput. Syst. (CHI)*, May 2011, pp. 11–20.
[36] C. Kison, O. M. Awad, M. Fyrbiak, and C. Paar, "Security implications of intentional capacitive crosstalk," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 12, pp. 3246–3258, Dec. 2019.

Mohammad Mahdi Bidmeshki received the B.Sc. and M.Sc. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2004 and 2006, respectively, and the Ph.D. degree in computer engineering from The University of Texas at Dallas, Richardson, Texas, in 2018. He is currently a Postdoctoral Research Associate with The University of Texas at Dallas. His current research interests include hardware-based security, trusted hardware design, formal methods in security and verification, and the applications of machine learning in computer security.

Angelos Antonopoulos (Member, IEEE) received the M.Eng., M.Sc., and Ph.D. degrees from the School of Electronic and Computer Engineering, Technical University of Crete, Chania, Greece, in 2005, 2008, and 2014, respectively. In 2015 he joined the Trusted and RELiable Architectures (TRELA) Research Laboratory, The University of Texas at Dallas (UTD), Richardson, TX, USA, as a Postdoctoral Research Associate. He is currently a Senior Patent Engineer with u-blox Athens S.A., Greece. His research interests include the design of trusted and reliable analog/RF integrated circuits and systems, hardware security in wireless networks, and design-oriented compact modeling of advanced semiconductor devices. He was a recipient of the Best Hardware Demonstration Award from the 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST'18).

Yiorgos Makris (Senior Member, IEEE) received the Diploma degree in computer engineering from the University of Patras, Greece, in 1995, and the M.S. and Ph.D. degrees in computer engineering from the University of California at San Diego in 1998 and 2001, respectively. After spending a decade with the Faculty of Yale University, he joined the UT Dallas where he is currently a Professor of electrical and computer engineering leading the Trusted and RELiable Architectures (TRELA) Research Laboratory, and the Safety, Security, and Healthcare Thrust Leader of the Texas Analog Center of Excellence (TxACE). His research interests include applications of machine learning and statistical analysis in the development of trusted and reliable integrated circuits and systems, with particular emphasis in the analog/RF domain. He was a recipient of the 2006 Sheffield Distinguished Teaching Award, the Best Paper Award from the 2013 IEEE/ACM Design Automation and Test in Europe (DATE'13) Conference and the 2015 IEEE VLSI Test Symposium (VTS'15), and the Best Hardware Demonstration Award from the 2016 and the 2018 IEEE Hardware-Oriented Security and Trust Symposia (HOST'16 and HOST'18). He serves as an Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and the *IEEE Design & Test of Computers*. He served as a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.