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

Mohammad-Mahdi Bidmeshki, Angelos Antonopoulos, and Yiorgos Makris
Department of Electrical Engineering, The University of Texas at Dallas
Richardson, Texas 75080
Email: {bidmeshki, aanton, yiorgos.makris}@utdallas.edu

***Abstract*—Information flow tracking (IFT) is a widely used methodology for ensuring data confidentiality in electronic systems and numerous 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). The risk of accidental information leakage, however, is not restricted to the digital domain. Indeed, analog signals originating from sources of sensitive information, such as biometric sensors, as well as analog outputs of a circuit, could carry or leak secrets. Moreover, similar to digital designs, analog circuits can also be contaminated with malicious information leakage channels capable of evading traditional manufacturing test. Compounding the problem, in analog/mixed-signal circuits such information leakage channels can cross the analog/digital or digital/analog interface, making their detection even harder. To this end, in this paper we introduce a PCHIP-based methodology which enables systematic formal evaluation of information flow policies in analog/mixed-signal designs. As we demonstrate, by integrating IFT across the digital and analog domain, our method 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.**

I. Introduction

Information flow tracking (IFT) [1] was introduced as a powerful approach for enforcing information flow policies in computer systems. More specifically, its original objective was to verify the confidentiality and/or integrity of sensitive data, by ensuring that such data does not get contaminated and/or does not reach unauthorized sites. Since its introduction, IFT has been applied in multiple contexts and has been implemented at various levels. In its basic but fundamental form, IFT augments each data element with sensitivity tags. Information flow policies are, then, used to define rules on propagating and manipulating these sensitivity tags in accordance with the operations performed on their corresponding data elements.

At the software level, static IFT methods enforce information flow policies on a program at compile-time based on logical inference and reasoning [2]. 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 [3]. 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 [4]. For example, the gate-level IFT approach proposed in [5] refines the conservative rules used by higher-level IFT methods and implements more realistic tag computation operations.

At the hardware level, the need for IFT methods has only recently arisen, due to the globalization of the integrated circuit (IC) supply chain and the concomitant trustworthiness concerns. Specifically, while enforcing information flow policies, software and system-level IFT methods consider the underlying hardware as trusted. However, as the various stages of contemporary IC design and fabrication involve multiple parties and are distributed across the globe due to economic considerations, this assumption is being challenged. Indeed, the threat of malicious attacks wherein adversaries may conceal hardware Trojans or embed back-doors in the design of an IC for future exploitation is now considered realistic [6], [7]. As a result, among the various hardware evaluation methodologies developed to address the trustworthiness concern, hardware-level IFT has also been investigated [8], [9]. The basis for such hardware-level IFT was the proof-carrying hardware intellectual property (PCHIP) framework [10], wherein formal proofs of security properties accompany the hardware description language (HDL) code of a module, in order to prevent introduction of malicious functionality. In PCHIP-based IFT, such security properties are specifically crafted to prevent sensitive information leakage.

A key limitation of existing PCHIP-based hardware IFT methods is that they are applicable only to the digital domain. However, confidentiality and integrity of sensitive data may also be jeopardized in the analog domain or in mixed-signal designs. Indeed, previous studies have demonstrated how secret information from the digital domain can be leaked via systematic modification of design parameters in the analog domain [11], such as changing carrier frequencies or transmission power in an RF transmitter. The problem intensifies as mixed-signal IC designs become widespread due to ubiquitousness of wireless technologies, such as Wi-Fi and Bluetooth, and as simple digital I/Os of the past are being replaced by high-speed links which extensively combine analog and digital techniques for noise reduction or channel distortion compensation [12]. Indeed, as the complexity of mixed-signal designs increases, adversaries are afforded many opportunities for implementing such malicious functionality, sometimes as simple as adding one extra transistor, without violating acceptable design specifications. Therefore, developing an IFT approach for analog/mixed-signal designs would greatly assist in detecting potential information leakage paths and establishing trustworthiness in such designs.

To this end, in this paper we introduce an IFT method capable of seamlessly crossing the analog/digital boundary. Specifically, by extending the PCHIP-based IFT of the digital domain to the transistor level, we create a unified framework for enforcing information flow policies in digital, analog, and mixed-signal designs. Various other approaches have been proposed in the past for formally reasoning on an analog/mixed-signal design. For example, Booleanization of analog circuits

978-3-9815370-8-6/17/$31.00

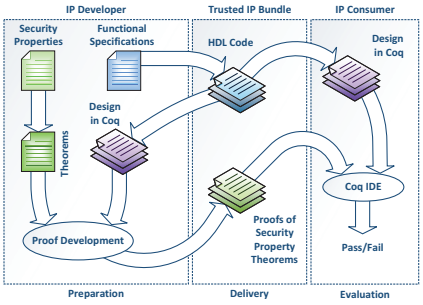

Fig. 1. PCHIP framework [10], [15]

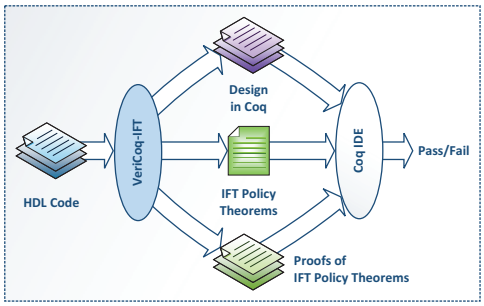

Fig. 2. Automated PCHIP framework for information flow policies [9]

has been utilized for formal state exploration of design characteristics [13]. Similarly, an effort to create a framework for formal verification of analog/mixed signal designs based on symbolic computation was presented in [14]. To the best of our knowledge, however, this work is the first attempt to develop an IFT method for analog/mixed-signal circuits.

The rest of this paper is organized as follows. Section II reviews the digital PCHIP-based IFT framework. Section III introduces the proposed IFT extension for analog designs. Section IV describes its integration with the digital PCHIP-based IFT framework. Section V demonstrates our method's ability to reveal sensitive information leakage in two analog/mixed-signal designs. Conclusions are drawn in Section VI.

II. 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 [10], which is shown in Fig. 1, 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 [15], such as Coq [16], 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 [17].

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 [8], [9] 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 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]. As shown in Fig. 2, *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.

To elaborate further, consider the Verilog source code of Fig. 3 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. To determine this value, designers need to consider the number of sensitivity reducing operations the signal should go through, before reaching the output [9]. The special comment in line 21 defines the eXclusive-OR operation of line 22 as a sensitivity reducer.

Fig. 4 shows the Coq representation produced by *VeriCoq-IFT* for the Verilog source code of Fig. 3. 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 **`Some`** `1`. Since there is no annotation of initial sensitivity level for other signals in the Verilog source code of Fig. 3, **`None`** is used for their sensitivity level in the list of lines 31-36 in Fig. 4.

Lines 3-20 in Fig. 4 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. 4 evaluates the code in the Coq representation based on the

```
module ift_sample (out, clk, inp, k);
  output [1:0] out;
  input clk;

  /* vericoq init_sensitivity_level_1 */
  input [1:0] inp;
  input [1:0] k;
  reg [1:0] out;
  wire a, b;

  always @(posedge clk)
    out <= {a, b};
  assign a = inp[1] & k[0];
  my_op op1 (.out(b), .i1(inp[0]), .i2(k[1]));
endmodule

module my_op (out, i1, i2);
  output out;
  input i1, i2;

  /* vericoq sensitivity_reducer */
  assign out = i1 ^ i2;
endmodule
```

Fig. 3. An example Verilog source code

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. 4 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.

III. 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 [8], [9] 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.

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 extract high-level analog modules (e.g. amplifiers, mixers, etc.) from the transistor level design and perform IFT at the analog module level. In order to gain a fundamental understanding of analog signal flows at the finest granularity, in this work we chose to implement analog IFT at the transistor level, deferring other options to our future research. 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. 5, 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. 6, 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.

```
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. 4. Coq representation, security property theorem and proof generated by *VeriCoq-IFT* for Verilog code of Fig. 3

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.

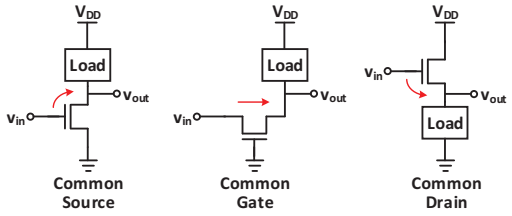

Fig. 5. MOSFET configurations in amplifiers and possible data flows

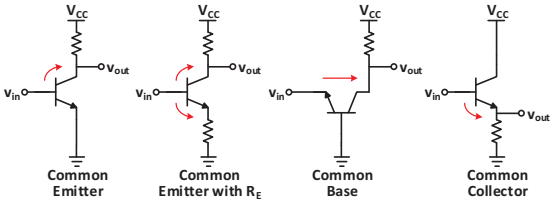

Fig. 6. BJT configurations in amplifiers and possible data flows

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. Therefore, nodes connected to the power supply cannot carry sensitive data in our information flow model. Based on these rules, Fig. 7 shows the data flow as it progresses from the input to the output of an amplifier through several components.

In cases where a MOSFET is used as a capacitor, its source, drain and bulk terminals are connected together and form one terminal of the capacitor. The gate of the transistor serves as the other terminal of the capacitor, as shown in Fig. 8. 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. 7, such a flow may exist and may be missed by our model. Indeed, as shown in Fig. 8, 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 connected together, with a capacitor.

IV. 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*, reviewed in Section II, 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 be evaluated in Coq. To take advantage

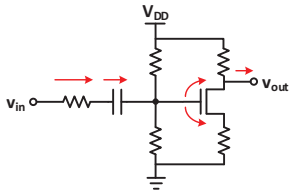

Fig. 7. Data flow from the input to the output of an amplifier through resistor, capacitor and NMOS transistor

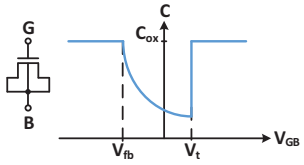

Fig. 8. MOSFET used as a capacitor at various voltages [18]. V_t and V_{fb} are threshold and flat-band voltages respectively

of this framework for our purpose, we need to: a) have a Verilog netlist of the analog/mixed-signal design, and b) define transistor-level analog data flow policies in *VeriCoq-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 (RT) or the gate level, and the analog part at the transistor 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 III, 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. 9 shows sample module definitions for modeling IFT in capacitors, as well as in NMOS and NPN transistors. 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 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.

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

```
// Modeling analog data flow in capacitors
// Resisitors, 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
```

Fig. 9. Sample module definitions to mimic analog data flows in *VeriCoq-IFT*

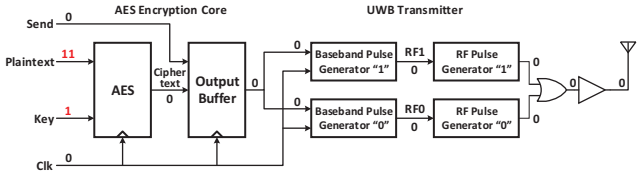

Fig. 10. The block diagram of wireless cryptographic IC design [11]. Numbers represent the propagated sensitivity levels

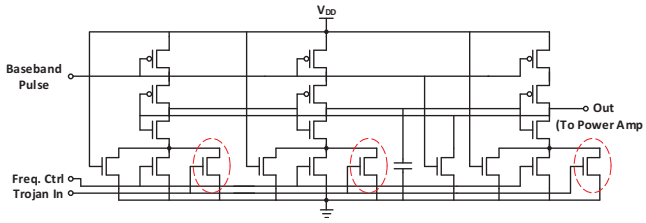

Fig. 11. Transistors added to enable information leakage through varying carrier frequencies in RF pulse generators

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.

V. 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].

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 [11], whose block diagram is shown in Fig. 10. Authors in [11] 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. 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, 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. 10 also shows the sensitivity levels propagated in this design.

[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.

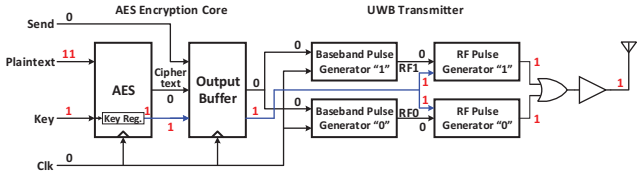

Fig. 12. Information leakage path in Trojan varying carrier frequency

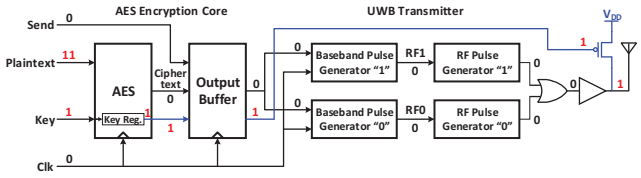

Fig. 13. A Trojan which adds a transistor in the power amplifier to leak information by varying transmission power, along with its leakage path

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. 11. By evaluating this design in the enhanced *VeriCoq-IFT*, we observe that the proof does not pass in Coq. This implies that a possible path exists through which sensitive information may leak to the output. For further insight, Fig. 12 shows the information leakage path and the propagated signal sensitivity levels in this design.

3) Transmission Power Trojan: 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, as shown in Fig. 13. Following the same procedure as in Section V-A1, 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. 13 also shows the information leakage path and the signal sensitivity levels in this design.

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. 14 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 heartbeat 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 [19]. Our example circuit borrows a technique called *A2*, which was recently introduced in [20] 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. 14, is activated. The

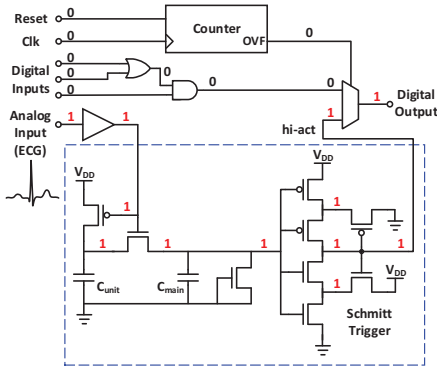

Fig. 14. An example circuit leaking information from the analog to the digital domain. The dashed box shows *A2*, which was introduced in [20]. Numbers represent the propagated sensitivity levels

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 in [20], 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 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 proposed approach, we followed a similar procedure as in Section V-A1 and converted the design to its corresponding Coq representation, while marking the analog input as sensitive. Verification of the proof of 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. 14 show the propagated sensitivity levels which lead to the digital output being marked as sensitive.

VI. Conclusion

We presented an IFT approach for analog designs and we integrated it into *VeriCoq-IFT*, an automated PCHIP-based framework for enforcing information flow policies on digital designs. Thereby, we developed what we consider as the first IFT solution which enables designers to seamlessly enforce information flow policies in analog/mixed-signal circuits. As we showed through example circuits, inadvertent design errors or malicious embedded capabilities leading to information leakage may be revealed by this extended framework, even when such leakage crosses the analog to digital boundary or vice versa. Besides applying our method to larger designs and fine-tuning our information flow models, our ongoing research focuses on raising the abstraction level at which we track analog/mixed-signal information flow above the transistor level, anticipating higher versatility of our method and better accuracy of our result.

Acknowledgment

This work was partially supported by the National Science Foundation (NSF 1318860) and the Army Research Office (ARO W911NF-12-1-0091).

References

[1] A. C. Myers and B. Liskov, "A decentralized model for information flow control," in *ACM Symposium on Operating Systems Principles (SOSP)*, 1997, pp. 129–142.

[2] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 5–19, 2003.

[3] L. C. Lam and T. Chiueh, "A general dynamic information flow tracking framework for security applications," in *Annual Computer Security Applications Conference (ACSAC)*, 2006, pp. 463–472.

[4] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas, "Secure program execution via dynamic information flow tracking," in *Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2004, pp. 85–96.

[5] M. Tiwari, H. M. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood, "Complete information flow tracking from the gates up," in *Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2009, pp. 109–120.

[6] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 6:1–6:23, 2016.

[7] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[8] Y. Jin, B. Yang, and Y. Makris, "Cycle-accurate information assurance by proof-carrying based signal sensitivity tracing," in *IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 99–106.

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

[10] E. Love, Y. Jin, and Y. Makris, "Proof-carrying hardware intellectual property: A pathway to trusted module acquisition," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 25–40, 2012.

[11] Y. Liu, Y. Jin, and Y. Makris, "Hardware Trojans in wireless cryptographic ICs: silicon demonstration & detection method evaluation," in *Int. Conf. on Computer-Aided Design (ICCAD)*, 2013, pp. 399–404.

[12] 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 Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 871–880, 2016.

[13] 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 *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 250–255.

[14] 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*. Springer, 2015, pp. 115–145.

[15] M.-M. Bidmeshki and Y. Makris, "VeriCoq: A Verilog-to-Coq converter for proof-carrying hardware automation," in *Int. Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 29–32.

[16] INRIA. The coq proof assistant. [Online]. Available: http://coq.inria.fr/

[17] Y. Jin and Y. Makris, "A proof-carrying based framework for trusted microprocessor IP," in *Int. Conf. on Computer-Aided Design (ICCAD)*, 2013, pp. 824–829.

[18] C. Hu, *Modern semiconductor devices for integrated circuits*. Prentice Hall, 2010.

[19] A. Raij, A. Ghosh, S. Kumar, and M. Srivastava, "Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment," in *SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 11–20.

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