Security Validation of VP-based Heterogeneous Systems: A Completeness-driven Perspective *

Ece Nur Demirhan Coskun[a], Muhammad Hassan[a,b], and Rolf Drechsler[a,b]
[a]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
[b]Institute of Computer Science, University of Bremen, 28359 Bremen, German

Abstract

The widespread use of modern, feature-packed heterogeneous systems has increased the need for robust security measures. A single bug may cause far-reaching and devastating consequences, blocking accessibility of various in-house and third-party *Intellectual Properties* (IPs) and resulting in an entire system's failure. In this regard, the concept of *Completeness-Driven Development* (CDD) provides the promise of early detection of bugs and an accelerated design process. The high-level idea is to use *Virtual Prototypes* (VPs) at the abstraction of *Electronic System Level* (ESL) as the starting point for early hardware/software co-design and verification. Going down the abstraction levels, the next abstraction in the design process can only be entered if completeness at the current abstraction level has been achieved. Completeness refers to checking whether the entire behavior of the design has been verified. However, CDD was introduced for functional verification of digital systems without considering security. In comparison, the modern systems are heterogeneous and security is of utmost importance. In this paper, we look at security validation of VP-based heterogeneous systems from the perspective of CDD. We provide an overview of the current state of security validation techniques and highlight the need for CDD to ensure the security. More concretely, we propose a novel *Information Flow Tracking* tool – VAST, for complex heterogeneous systems using SystemC-AMS VPs. VAST targets availability of IPs as the threat model. Our experimental results on real-world case-studies show the applicability and scalability of VAST.

1 Introduction

The integration of *Software* (SW), digital *Hardware* (HW) with microcontrollers and microprocessors, and *Analog/Mixed-Signal* (AMS) *Intellectual Property* (IP) has boosted the functionality of heterogeneous embedded systems widely used in *Internet of Things* (IOT) devices. However, as IoT devices store more private information and perform security-sensitive tasks, the need for higher security measures has risen in recent years. To ensure complete security validation in such systems, a holistic approach is necessary, covering not just the SW or digital HW but also the AMS IPs such as physical interfaces, sensors, and actuators [1, 2, 3].

In this regard, *Completeness-Driven Development* (CDD) [4] is often approached in which the design process is divided into different abstraction levels, consequently, shifting focus towards verification. The high-level idea is to use *Virtual Prototypes* (VPs) at the abstraction of *Electronic System Level* (ESL) as the starting point for early design and verification process, and it progresses to the next abstraction level only after achieving completeness, i.e verifying the complete behavior of the design at each level of abstraction.

While originally CDD was introduced for functional verification, recent advances in security validation methods in the digital domain using VPs [5, 6, 7, 8, 9] have been successfully employed early in the design phase. These methods use *Information Flow Tracking* (IFT) techniques to focus on digital hardware from various security concerns (e.g. confidentiality, integrity, and timing channel). IFT allows for understanding the flow of information across a system and identifies potential security flaws in design. In principle, IFT tags data objects to represent security classes, which have varying meanings based on the type of *Security Property* (SP) being analyzed and verifies information flow properties, enforcing rules for secure information flow, such as confidentiality, integrity, isolation, constant time, and availability. However, modern systems are heterogeneous with stringent security requirements.

In this paper we provide an overview of security validation of VP-based heterogeneous systems in the context of CDD. We propose a novel IFT tool – VAST, for complex heterogeneous systems using SystemC AMS VPs [10]. VAST uses a security property-based verification technique and analyzes given VPs to detect security vulnerabilities early in the design phase. We consider the availability problem as the threat model, which requires IPs to be readily accessible and usable for authorized users. In other words, if the availability of IPs is not guaranteed, it can cause system failures and block access to IPs. Our experimental results on real-world case-studies show the applicability and scalability of VAST.

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project AUTOASSERT under contract no. 16ME0117.

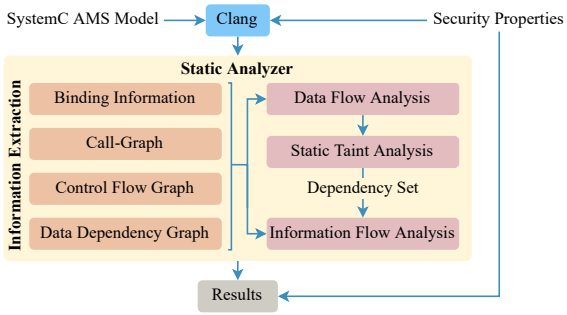

Figure 1 VAST overview for security validation of VP-based heterogeneous systems

2 Methodology

This section gives a general overview of VAST and its workflow. Figure 1 depicts the overall workflow of the tool to perform IFT analysis for heterogeneous systems. First a motivating example is provided to demonstrate the security threat model it addresses. Afterwards, a brief overview of VAST is provided.

2.1 Threat Model

Threat modeling plays a crucial role in ensuring the security of a system, as a single vulnerability can result in its compromise. In this paper, the considered threat model is "availability", i.e. timely accessibility, of IPs. In the context of heterogeneous systems, particular attention should be paid to securing assets involved in AMS interactions, such as sensors. No sensor input should be allowed to make the system unusable or unavailable. This can be done by ensuring the functionality of analog-to-digital and digital-to-analog interfaces and protecting sensitive data during transmission between different IP components.

2.2 Motivating Example

The motivating example is used to illustrate the principles of our methodology. The example is a simplified ECG monitoring device, which includes components such as an *Electrocardiography* (ECG) sensor, an *Analog IP* with *Schmitt Trigger* and *Analog Trigger* circuit, *Digital Controller*, *Bus*, and *Memory* as shown in Figure 2. The ECG sensor is connected to the *Schmitt Trigger* circuit before it is sent to the *Analog Trigger* circuit, which decides if the signal should be stored in Memory. When ECG signals surpass a set limit, the *Analog IP* will identify the irregularity and send a request to the *Memory* for storage via the *Bus*. The *Digital Controller* regularly carries out read and write actions in the *Memory*. To ensure availability, neither the *Digital Controller* nor the *Analog IP* should be able to block access to the *Memory*. The ECG monitoring system is implemented using SystemC-AMS.

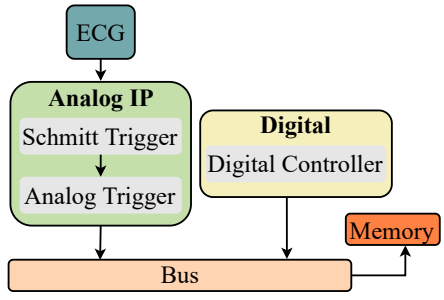

Figure 2 The SystemC AMS design of the motivating example – ECG monitoring device

Now consider a scenario that an access control policy has been implemented in the *Bus*. The highest priority is given to the *Analog IP*[1] through the use of a priority encoder. The detection of abnormal heart rate levels, for precautionary or emergency purposes, is made possible through this implementation. Due to the higher priority given to the *Analog IP*, the information flow between the *Digital Controller* and the *Bus* is indirect. This vulnerability can be exploited by an attacker to block the *Digital Controller*, making the *Memory* inaccessible. Detecting this type of direct/indirect information flow between IPs can be challenging, especially without advanced automated analysis techniques.

2.3 VAST - Static Information Flow Tracking Tool

In this section we briefly describe the functionality of VAST. VAST performs static information flow tracking using security properties provided as input and requires only a single execution to validate the security properties. It starts by reading the specified SPs and determining the critical signals. Then, it uses IFT to perform *Static Taint Analysis* (STA), followed by *Information Flow Analysis* (IFA) to validate the SPs.

2.3.1 Information Extraction

There are two main phases to perform the static information flow tracking: *Information Extraction* and *Static Analysis*.

The *Information Extraction* phase starts with an SP specification with respect to the security goal of the entire system. For example, if the goal is ensuring availability, i.e. various IPs are required to be available in a timely manner [11], SPs can be defined as follows:

$$SP = \{(SI, SO) | SI \leftarrow \{.. = HS\}, SO \leftarrow \{.. = AA\}\} \quad (1)$$

In Eq. (1), SP has inputs with the *High Security* (HS) tag and outputs that must be *Always Available* (AA) when needed.

For instance, an SP of the motivating example can be described as follows:

$$SP = (\{\texttt{st\_in} = HS\}, \{\texttt{grant\_digital} = AA\}) \quad (2)$$

In Eq. (2), the SP ensures that the signal `grant_digital` sent by the *Digital Controller* to the *Bus* module must not be dependent on the primary input `st_in` of the *Schmitt Trigger* module.

After the definition of SP, the *Binding Information* (BI) is extracted for the module connectivity. This information is

[1] It is a common practice to use such policies in interrupt controllers.

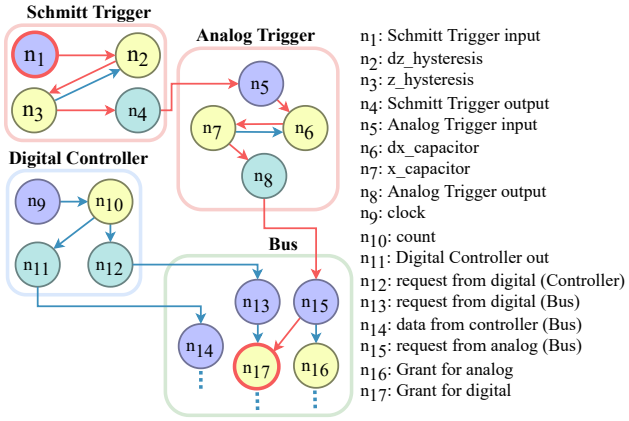

Figure 3 A part of extracted DDG of the ECG monitoring device

crucial for understanding how data moves through the system and for constructing *Call-Graphs* (CG). The method constructs the call-graph once at the beginning. The CG is used to coordinate the analysis so that the information is propagated to the correct function inside the AMS VP. As a result, it makes use of the BI to appropriately identify the function calls.

At the next step, *Control Flow Graph* (CFG) is extracted from the *Abstract Syntax Tree* (AST) of the VP to understand the relationship between various statements (data flow and control flow) of the design. Then, the *Data Dependency Graph* (DDG) is used to understand the relationship between the variables in a design, including signals, ports, and global and local variables of all modules [10]. It provides information on how these variables are connected to each other and how they interact in the design [5]. For example, in Figure 3, a part of the DDG of the motivating example is shown visually with purple-colored nodes representing input ports, yellow-colored nodes representing internal signals, and green-colored nodes representing output ports. The BI is abstracted away.

2.3.2 Static Analysis

Static Analysis consists of *Data Flow Analysis* (DFA), *Static Taint Analysis* (STA) and *Information Flow Analysis* (IFA) as shown in Figure 1. The DFA algorithm is used to identify the test objectives, i.e. definition-use (*def-use*), for a given VP by performing intra-function analysis. A procedure referred to as *reaching uses* is employed to determine these def-use pairs for a VP, which effectively addresses the question of "for each defined variable, which uses may potentially utilize its values?".

Next, the STA algorithm is performed to generate the Dependency Set. It starts with a tainted source and adds variables based on dependence data from the CFG, including *def-use* pairs and *use-to-dependence*. The *use-to-dependence* represents the dependence for variables in the conditional statements of CFG blocks, where definitions in the possible succesors to the conditional statements are stored. For example, in Figure 3, we aim to show clearly the critical path from the *Schmitt Trigger* input to the grant digital with red arrows, which consist of nodes in the *Dependency Set*.

At the end, the IFA algorithm is applied to identify all potential information flows in an AMS VP by using the DDG, the CFG, and a set of SPs. The IFA algorithm traces forward from secure input nodes in the DDG to the output nodes (AA tag), to find out whether a variable is affected by secure inputs (HS tags). The affected variables are added to a sensitive list of secure inputs SL_{SI}. Additionally, a backward tracing on the DDG is performed to find out whether the final output receives its final value via intermediate variables. Hence, the algorithm extracts the variables of assignment statements that are explicitly or implicitly related to the outputs with the AA tag. These nodes are added to the sensitive list of secure outputs SL_{SO} along with the corresponding design variables. Then, the CFG is analyzed to find sensitive control signals that affect the occurrence of updates on variables with AA tags. The algorithm extracts control variables from each condition node in the CFG, such as if-else statements, and performs further analysis if there is an overlap with the sensitive list of secure inputs. The purpose of this analysis is to find assignment statements where the left-hand side variables are in the list of secure outputs *SO* for explicit flow or the list of sensitive secure outputs SL_{SO} for implicit flow. If any flow exists in the design, the condition and assignment nodes in the CFG are reported back to the user.

3 Experimental Results

To evaluate the effectiveness of VAST on complex heterogeneous systems, a car anti-trap window system (26604 line of codes) is used [10]. In this experiment, the system regulates the window's movement (up and down) while preserving the safety of the passengers. When an obstacle such as a passenger's finger is detected, the current flow changes, notifying the controller to halt and prevent hazards.

Regarding availability security concern, the detection of the obstacle must be independent from unrelated signals such as the up/down control buttons that change the movement of the car window. In such a system, this security breach may result in an attacker being able to take control of the system. To assess the security of the design against this attacker, we have defined five availability SPs. VAST found that one of these SPs was unsatisfied, which is given in Eq. 3.

$$SP = (\{up\_key\_o = HS\}, \{obstacle\_detected\_o = AA\}) \quad (3)$$

Eq. (3) checks if the flow of the *obstacle_detected_o* signal to the ECU is dependent on the other signals. According to the SP, the signals that are used for the detection of the obstacle, such as the *obstacle_detected_o* signal, must be available in the ECU regardless of the values of any unrelated signal, such as the *up_key_o* signal sent by the corresponding module. VAST found that the variable *clamping_protection_current_state* was dependent to *up_key_o*. Consequently, this variable was a controlling variable of a switching block that affects *obstacle_detected_o*, and caused the SP to fail. The experiment was carried out on a PC equipped with 24 GB RAM

and an Intel Core i7-8565U CPU running at 1.8 GHz. The run time for this example was 167.2 s.

4 Discussion and Future Work

One possible direction is to investigate the transferability of the SPs. We believe that the proposed methodology should be extended/expanded as per CDD concepts by performing potential vulnerability analysis in subsequent abstraction levels to cover all vulnerabilities. This may deepen our understanding of the transferability of verified SPs, with respect to the CDD flow [4].

Additionally, suitable completeness measures are needed for each abstraction level to check the entire behavior of the design. In this regard, security metrics and coverage metrics need to be explored. Furthermore, even if the security metrics of the design are satisfied for an abstraction level, going down to a lower level abstraction may introduce new vulnerabilities. These are called "the transformation-induced" changes in the design while moving between various abstraction levels (e.g., SystemC/RTL/Gate/Layout). They can expose the design to additional vulnerabilities. In this regard, the security properties should be adapted for the next abstraction level, e.g., from SystemC AMS to Verilog AMS or gate-level, new signals may appear in the design. This requires an additional property definition to maintain the security goal same as the security goal at the system-level. As a next step, we aim to explore techniques to assist accurate transformation and SP refinement.

5 Conclusion

We have introduced a novel VP-based IFT tool against availability security properties for heterogeneous systems in the context of CDD. VAST utilizes static information flow tracking that operates directly on the SystemC-AMS VP models. It includes data flow analysis and taint analysis to identify static paths that violate specified availability properties. Vulnerabilities of the design are reported back to the user for further inspection. The effectiveness of VAST has been shown in real-world systems.

6 Literature

[1] I. Polian, "Security aspects of analog and mixed-signal circuits," in *2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW)*, Jul. 2016, pp. 1–6.

[2] A. Antonopoulos, C. Kapatsori, and Y. Makris, "Trusted Analog/Mixed- Signal/RF ICs: A Survey and a Perspective," *IEEE Design & Test*, vol. 34, no. 6, pp. 63–76, Dec. 2017.

[3] M. Elshamy, "Design for security in mixed analog-digital integrated circuits," Ph.D. dissertation, Sorbonne université, 2021.

[4] R. Drechsler, M. Diepenbeck, D. Große, U. Kühne, H. M. Le, J. Seiter, M. Soeken, and R. Wille, "Completeness-Driven Development," in *Graph Transformations*, ser. Lecture Notes in Computer Science, H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, Eds. Berlin, Heidelberg: Springer, 2012, pp. 38–50.

[5] M. Hassan, V. Herdt, H. M. Le, D. Große, and R. Drechsler, "Early SoC security validation by VP-based static information flow analysis," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2017, pp. 400–407.

[6] M. Goli and R. Drechsler, "ATLaS: Automatic Detection of Timing-based Information Leakage Flows for SystemC HLS Designs," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2021, pp. 67–72.

[7] M. Goli, M. Hassan, D. Große, and R. Drechsler, "Security validation of VP-based SoCs using dynamic information flow tracking," *it - Information Technology*, vol. 61, no. 1, pp. 45–58, Feb. 2019.

[8] M. Goli and R. Drechsler, "Early SoCs Information Flow Policies Validation using SystemC-based Virtual Prototypes at the ESL," *ACM Transactions on Embedded Computing Systems*, Jun. 2022.

[9] P. Pieper, V. Herdt, D. Große, and R. Drechsler, "Dynamic Information Flow Tracking for Embedded Binaries using SystemC-based Virtual Prototypes," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6.

[10] E. N. Demirhan Coskun, M. Hassan, M. Goli, and R. Drechsler, "VAST: Validation of VP-based Heterogeneous Systems against Availability Security Properties using Static Information Flow Tracking," in *24th International Symposium on Quality Electronic Design (ISQED'23)*, Apr. 2023.

[11] E. Jonsson, "Towards an integrated conceptual model of security and dependability," in *First International Conference on Availability, Reliability and Security (ARES'06)*, Apr. 2006, pp. 646–653.