Run-time Self Attestation of FPGA Based IoT Devices

Muhammad Usama, *Student Member, IEEE*, Muhammad Naveed Aman, *Senior Member, IEEE*, and Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—Flexibility and reconfigurability make FPGAs ideal for IoT applications because they enable efficient customization and optimization of hardware acceleration tasks in diverse IoT applications. Malicious hardware trojans pose a significant security threat, capable of compromising the integrity of reconfigurable devices such as FPGAs. The majority of current attestation schemes either demonstrate complexity and demand significant resources or lack versatility. To solve this issue, this paper proposes a novel lightweight run-time attestation approach to detect hardware trojans or malicious modifications in a hardware design. The proposed technique can verify the integrity of both the hardware design's finite state machine and its datapath. Attesting the finite state machine ensures the accuracy of state transitions and control behavior while verifying the datapath validates the data processing operations. When combined, these provide a comprehensive validation of the overall hardware functionality. A trusted verifier initiates challenges by stipulating a starting state and an input sequence to the prover. The prover then executes these challenges and reports the observed responses, i.e., state transitions, control outputs, status outputs, and timing metrics. Anomalies between the expected and observed behaviors serve as indicators of potential trojan interventions. The proposed method's efficacy is substantiated through simulation and implementation on a Zynq-7000 SoC, showcasing its efficiency in terms of resource utilization overhead. Collectively, this study advances the capabilities of remote attestation while bolstering the security of reconfigurable platforms.**

***Index Terms*—Attestation, datapath, FPGA, FSM, hardware security, hardware trojan**

I. Introduction

A Field-Programmable Gate Array (FPGA) device consists of a two-dimensional array of configurable logic blocks (CLBs), which can be interconnected through routing channels to implement a digital logic circuit. A hardware description language (HDL), such as Verilog or VHDL, is used to describe the design of a digital circuit. HDLs allow designers to program, modify, and reprogram the hardware design of digital circuits. The popularity of FPGA devices has increased significantly because of the advantages these devices provide as compared to general-purpose microprocessors and/or Application-Specific Integrated Circuits (ASICs) such as parallel processing capability, reconfigurability, higher power efficiency, Shorter time to market, and lower upfront non-recurring engineering cost [1]. Dynamic partial reconfiguration is another notable feature found in modern FPGA devices, enabling the reconfiguration of specific portions while the remaining sections of the FPGA remain active and operational. This capability offers extensive versatility by allowing a diverse array of computational units to be utilized within a single device [2]. Nevertheless, the dynamic reconfigurability inherent in FPGA devices also presents potential avenues for malicious actors to exploit, introducing hardware trojans or altering the original design with the intent to disrupt the intended functionality of these devices. These potential attacks can compromise the security and reliability of the system resulting in critical data theft, operation disruption, unauthorized access, etc [3].

Cyber vulnerabilities highlight the importance of devising security measures and rigorous validation procedures to guarantee the reliability and integrity of FPGA configurations; thereby, protecting against potential attacks and guaranteeing the secure operation of FPGA devices (especially for sensitive applications). In general, attestation mechanisms are employed to verify the software/firmware integrity of a target device. The purpose of attestation is to establish a mechanism wherein an untrusted device undergoing testing can demonstrate to a remote trusted verifier that the executed code remains unaltered. A generic attestation protocol is depicted in Figure 1 [4]. The verifier initiates the process by sending a challenge to the device, also known as the prover. Subsequently, the prover executes a local algorithm to compute a value, which is then transmitted back as a response. By evaluating the received value the verifier assesses whether any modifications were made to the device's intended design.

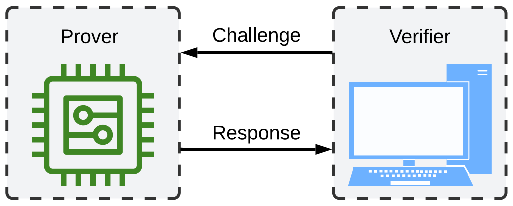

Fig. 1: A generic attestation protocol.

This research was supported in part by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research Development Programme, under grant FCP-NUS-RG-2022- 019. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

M. Usama and M. N. Aman are with the School of Computing, University of Nebraska-Lincoln, 1400 R St, Lincoln, Nbraska, United States 68588. Email: usama2@huskers.unl.edu, naveed.aman@unl.edu.

B. Sikdar is with the Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117583. Email: bsikdar@nus.edu.sg.

Various hardware attestation techniques have been proposed to secure FPGA devices against a range of hardware attacks. However, the existing methods are either based on complex computation algorithms [1], [5]–[9] or limited to specific components of hardware design [10]–[12]. To solve these issues, this paper proposes a novel approach aimed at attesting the integrity of hardware designs through a meticulous examination of both the operation of the finite state machine (FSM) and the behavior of the datapath. Attestation of the operation of an FSM encompasses the validation of the state transitions and their associated control signals. In parallel, a comprehensive assessment of the datapath's behavior involves validating its response to control signals generated by the FSM, coupled with an evaluation of the response time to generate the anticipated responses. By comparing projected state transitions, the control signals generated by the FSM for each state, the status signals of the datapath in response to control signals, and the temporal metrics of each state against the tangible behavior observed during the attestation process, a comprehensive evaluation of the integrity of a prover's hardware is established. This methodology proves instrumental in pinpointing and mitigating potential vulnerabilities arising from hardware-based attacks, effectively enhancing the overall security of FPGA devices. The major contributions of this paper are as follows:

- An attestation approach for verification of the FSM in a hardware design against unauthorized modifications in state transition and/or control logic.
- An attestation approach for verification of hardware design of the datapath against malicious modifications.
- Experimental validation of the proposed technique on actual hardware, and comparison with existing techniques.

II. Literature Review

Many schemes have been proposed to protect FPGA devices from hardware trojans. The authors in [5] discussed hardware trojan attacks on FPGA devices and proposed an approach called Adapted Triple Modular Redundancy (ATMR) to protect FPGA devices against them. However, this approach requires extra resources as it utilizes three spare copies of the original hardware circuit to contain or bypass the effects of hardware trojans. An approach to detect hardware trojans in cryptographic intellectual property (IP) modules provided by an untrusted third party is proposed in [6]. This approach compares the optical microscopic pictures of the silicon product to the original view from a layout database reader. However, this approach fails to detect hardware trojans inserted in the RTL layer of a hardware design. Moreover, this approach only detects hardware trojans in cryptographic IP modules and does not guarantee the protection of the whole hardware design. The authors in [10] analyzed the detection of hardware Trojans in Scalable Encryption Algorithm (SEA) crypto. They proposed an approach to use path delays to detect hardware trojans inserted at different stages in the ASIC design flow at both the gate and layout levels. However, this method is limited to identifying hardware trojans that were introduced during the design and production phases. Furthermore, this strategy may not be utilized for other algorithms or technologies because it was developed for a specific technology (90nm libraries) along with a specific algorithm (SEA crypto). To protect the design against malicious hardware insertion, the authors of [7] developed a hybrid design verification technique. The unused circuit identification (UCI) technique is employed in the proposed scheme to automatically identify and delete harmful circuitry during the design verification stage. An adversary could, however, possibly bypass the UCI method by inserting malicious test cases that perform incorrect state checks.

The authors in [8] proposed an attestation scheme for reconfigurable devices by employing a cryptographic hash function to validate a bitstream. The authors also utilized delimitation of the area to optimize the attestation process. The authors in [1] proposed a scheme for the self-attestation of configurable hardware. This approach employed a hashing function-based message authentication code (MAC) to perform validation of the bitstream loaded into the configuration memory of an FPGA device. The authors in [9] proposed a secure architecture to remotely compute and deliver the code integrity attestation of an executing program. However, all these approaches require extensive computational resources as well as time-consuming operations such as cryptographically secure hash and/or MACs. The authors in [11] and [12] proposed protocols to prevent a remote update of malicious code on FPGA devices. These approaches provide security against replay attacks by ensuring the confidentiality and integrity of the hardware loaded into an FPGA device. However, these techniques only consider remote attacks and assume that physical attacks are not feasible.

Each of the above attestation approaches is either complex and demands extensive resources or targets a specific attack or part of hardware design. In contrast, the proposed attestation mechanism does not require extensive resources and possesses the generality to attest all types of hardware designs that are controlled by an FSM. A comparison between existing approaches and the proposed work is drawn in Table I.

TABLE I: Summary of existing techniques and comparison with the proposed work.

	[1]	[5] [7] [8] [9]	[6]	[10]	[11] [12]	Proposed
Resource-efficient	✗	✗	✗	✓	✓	✓
Detection of remote attacks	✓	✓	✗	✗	✓	✓
Detection of local attacks	✓	✓	✓	✓	✗	✓
Full design coverage	✓	✗	✓	✓	✓	✓
Run-time detection	✓	✓	✗	✗	✓	✓
Generality	✓	✓	✓	✗	✓	✓

III. Background, Models, and Assumptions

A. Background

1) Finite State Machine: FSMs are utilized for modeling and controlling the behavior of hardware designs. They organize the operation of a design into a finite set of states, define the actions/outputs associated with each state, and specify the transition mechanism between states. The state of an FSM refers to a particular configuration of the system at a given time and is based on the input it has received and the actions it has taken thus far. Action in an FSM refers to a defined activity to be executed at a specific time, exerting an influence on the system's behavior. A transition describes how a state can be changed based on the occurrence of a certain input. It denotes the connection between two states and lists the prerequisites for the FSM to change from one state to the next. An FSM can be formally defined as a sextuple as given in Equation (1) [13].

$$M = (Q, \Gamma, O, \delta, \lambda, q_0) \tag{1}$$

where

- Q: Set of finite states in the FSM.
- Γ: Set of finite possible inputs to the FSM.
- O: Set of finite possible outputs of the FSM.
- δ: Transition function, that maps a state and an input to the next state; $\delta : \Gamma \times Q \mapsto Q$.
- λ: Output function, that maps a state and an input to the corresponding output; $\lambda : \Gamma \times Q \mapsto O$.
- q_0: Initial state.

FSMs are usually divided into two categories: Mealy machines and Moore machines. For the Mealy machine, the output function λ is dependent on both the state and input, i.e., $\lambda : \Gamma \times Q \mapsto O$. Conversely, the output function λ only depends on the state of a Moore's machine, i.e., $\lambda : Q \mapsto O$.

2) Dynamic Partial Reconfiguration of FPGAs: The partial reconfiguration feature of modern FPGA devices enables designers to logically divide the hardware design into multiple partitions. Each of these partitions can be configured and reconfigured separately at run-time without affecting the regular operation of other partitions. These partitions can be classified into static and dynamic partitions. Static partitions usually remain unchanged and are never reconfigured. These partitions are used to contain the configuration for external interaction and control parts of the hardware design. Whereas, dynamic partitions contain the intended hardware design and can be reconfigured at any time. To reconfigure the dynamic partition of an FPGA, a bitstream that targets the specific dynamic partition is updated while the rest of the bitstreams remain unchanged. This type of bitstream is referred to as a partial bitstream. The proposed architecture utilizes the concept of partitioning to isolate the attestation part from the system design part.

B. System and Adversary Model

1) System Model: The system model consists of the prover (*Pr*), verifier (*Ve*), and a controller (*Co*). We assume that these entities have a pre-established secure communication channel over the public internet. The *Pr* is an FPGA device that has some specific set of resources and performs an intended operation when configured. The *Pr* is divided into secure (*SecPart*) and open (*OpenPart*) partitions using the partial configuration feature. The *SecPart* is a static partition to include the secure configuration. Whereas, the *OpenPart* is a dynamic partition that contains the intended hardware design, and can be reconfigured locally as well as remotely. The *Pr* also has a nonvolatile memory element (NVM) to store the configuration when power goes out. The *Ve* is a computer or a server and is not constrained in terms of computational resources. The *Ve* is responsible for initiating and conducting the attestation process. The *Co* is also a computer or a server that could be the same as *Ve* or a different entity. The *Co* can reconfigure the *SecPart* as well as the *OpenPart*. It also communicates the intended update in the hardware design to the *Ve*. The system model considered in this paper is depicted in Figure 2.

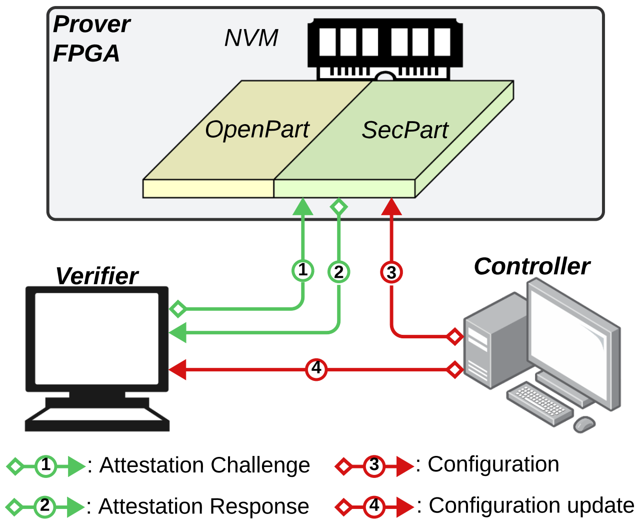

Fig. 2: System model.

2) Adversary Model: In this paper, we consider an adversary (*Ad*) that is capable of maliciously altering the configuration of a hardware design by injecting hardware trojans. We specifically concentrate on stealthy hardware trojans that target the flow of operations. We also assume that the adversary is capable of launching physical attacks that can modify the configuration of the hardware design to disrupt the operation flow. Other attacks that only target configuration memory and do not apply changes to the hardware design to disrupt the operation flow are excluded from our adversary model.

C. Assumptions

We make the following assumptions for this work

- Initially, both the *SecPart* and *OpenPart* of an FPGA are securely configured with the intended bitstream.
- The *SecPart* in the *Pr* can only be reconfigured by the *Co* and is protected against any modification attack.

- The NVM is a secure element and is only accessible from within the *Pr* for read and write operations. It has sufficient capacity to store all the configuration files.
- Both the *Ve* and the *Co* are considered trusworthy, impervious to any cyberattack.
- Communication channel between the *Co*, the *Ve*, and the *Pr* is secure to ensure the confidentiality and integrity of the communication.

IV. Proposed Technique

FSMs in the digital designs are tied to clock signals, synchronizing their operation with the system clock. These FSMs have well-defined finite states and transitions between states. Due to this deterministic nature, the *Ve* is required to validate the design only for scenarios that are explicitly defined in the design. This characteristic loosens the requirement of creating a large number of challenges, which avoids unnecessary complexity and system overhead. The deterministic nature of digital designs implies that the *Ve* can concentrate on specific and significant cases, in contrast to probabilistic systems where thorough testing might necessitate an extensive list of challenges. A *Ve* equipped with the knowledge of the FSM and the datapath functionality can choose challenges that cover all relevant states, transitions, and datapath activities. This focused strategy not only simplifies the attestation process but also improves its efficacy and effectiveness.

The proposed attestation mechanism is centered around validating the behavior of both the FSM and the datapath of a hardware design. The operation of the FSM is validated in run-time by analyzing its various parameters in Equation (1) that includes states (Q), transitions between states (δ), and the control signals (O) generated by the FSM in response to a given input status sequence (Γ). Whereas, run-time validation of the datapath is done by analyzing its response and response time to the control signals. A successful attestation outcome indicates the absence of any unauthorized modifications in the hardware design. In contrast, if a modification attack is attempted on the hardware design, it would disrupt the operation flow of the FSM and/or datapath, leading to a failed validation during the attestation process.

Since the proposed attestation technique is based on the validation of the design's functional behavior and is capable of dealing with low-level implementation modifications. The scheme is robust to implementation variations because it focuses on capturing the underlying functions of the digital design rather than relying on specific implementation details that may vary during the synthesis and compilation processes. As long as the critical structure and features of the design, including control and status signal flow, are retained, the proposed scheme can validate the design's integrity across many FPGA targets.

A. Proposed Architecture

In this section, we present a system architecture to complement the proposed attestation mechanism. This architecture mainly encompasses the following elements:

1) The Prover: We exploit the partial configuration feature of FPGAs by dividing the resources of the *Pr* into an *OpenPart* and a *SecPart* [14]. Moreover, an NVM is also included in the architecture of the *Pr* as described in Section III-B.

a) NVM: The integration of NVM in reconfigurable devices is a fundamental and widely adopted practice that offers numerous advantages, including reliable configuration persistence across power cycles, instant-on functionality, enabling in-system configuration updates, and optimizing power consumption [15]. Various protocols have been suggested and are utilized to ensure the security and smooth operation of NVMs [16]–[19]. Our proposed architecture involves the utilization of an NVM either as an internal component within the *Pr* or as an external module to store the configuration files of the design. Regardless of its location, we assume that the NVM is secure and only accessible for read and write operations from within the *Pr* [20]. Since most of the commonly used FPGAs are based on SRAM, their configurations are erased once the power is cut off. The updated design configuration files for the hardware design are always stored in the NVM. These design configuration files are loaded into the configuration memory of the *Pr* to reconfigure it with the design. We assume that the NVM's capacity is sufficient to store the entire configuration file, providing ample space for accommodating the complexities of the hardware design.

b) Secure Partition: The *SecPart*, constitutes a safeguarded area within the system, accessible solely to the *Co* for reconfiguration purposes. Within this partition, reside essential modules including communication, partial reconfiguration, and attestation modules. The partial reconfiguration module, residing in *SecPart*, assumes the responsibility of writing the configuration memory for the configuration/reconfiguration of the *OpenPart*. Furthermore, the communication module establishes vital connections, facilitating communication between the *Pr*, *Ve*, *Co*, and NVM, effectively bridging these components. Additionally, the attestation module, nestled within the *SecPart*, takes on the responsibility of computing the hardware design's response to challenges posed by the *Ve*. This process involves the *Ve* creating challenges for the *Pr* to authenticate itself. Upon receiving the challenge within *SecPart* of the *Pr*, the attestation module diligently computes the runtime response to the challenge. The internal architecture of the *Pr* is depicted in Figure 3.

c) Open Partition: The *OpenPart*, being openly accessible both locally and remotely, allows for dynamic configuration/reconfiguration from any location. This partition holds the primary hardware design deployed into the *Pr*. Due to its unrestricted accessibility and housing of the intended FPGA configuration, it becomes a primary target for potential adversaries. When presented with a challenge, the *SecPart* takes charge and performs the challenge execution on the *OpenPart*. The resulting response is then relayed back to the verifier, contributing to the system's overall attestation process.

2) Controller and Verifier: The *Co* is a desktop, laptop, or server. It holds central authority within the system and possesses full access to communicate, modify, and update

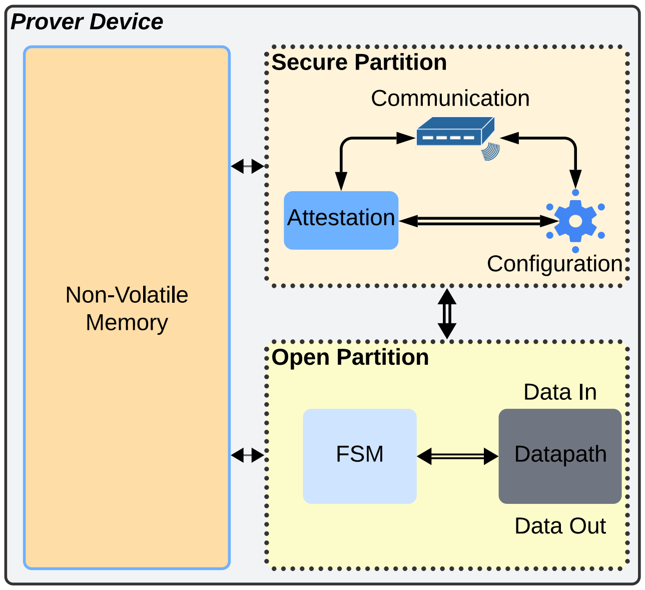

Fig. 3: A prover architecture.

all components. Specifically, the *Co* is exclusively authorized to make modifications to the hardware design within the *OpenPart* of the *Pr*. Additionally, it can alter the attestation module residing in its *SecPart*. Furthermore, the *Co* can update the (*Ve*) about any changes made to the hardware design. The (*Ve*), on the other hand, is an entirely independent system that can be deployed on a desktop, laptop, or server, or it can be an application running on the *Co* system. The primary responsibilities of the *Ve* include initiating the attestation process by generating a challenge and subsequently verifying the response received from the *Pr*. This verification is performed by comparing the response with the expected behavior, ensuring the integrity and security of the *Pr*.

B. Attestation Mechanism

The overall operation of the proposed attestation scheme is shown in Figure 4. The *Ve* generates a fresh challenge every time and dispatches it to the *Pr*. Subsequently, the *Pr* counterpart computes the response of the incoming challenge and sends it back to the *Ve*, where it is compared with the expected response to conclude the attestation process. The proposed attestation mechanism offers flexibility by introducing three distinct modes of attestation: Mode 1; targeting solely the FSM, Mode 2; concentrating solely on the datapath, and Mode 3; encompassing the overall hardware design as shown in Figure 4. The flexibility these various attestation modes provide enables the *Ve* to selectively execute attestation on either the FSM or the datapath independently. This segregation not only optimizes resource utilization but also streamlines the attestation process, making it efficient in terms of both resource consumption and execution time. Additionally, Our approach provides flexibility for the *Ve* to attest any specific portion of the design. For extremely large designs, the *Ve* can divide the entire design into smaller, manageable parts and attest each part separately. This incremental attestation ensures thorough verification while addressing scalability concerns. By breaking down the attestation process, the *Ve* can handle large designs effectively without compromising the integrity and completeness of the attestation.

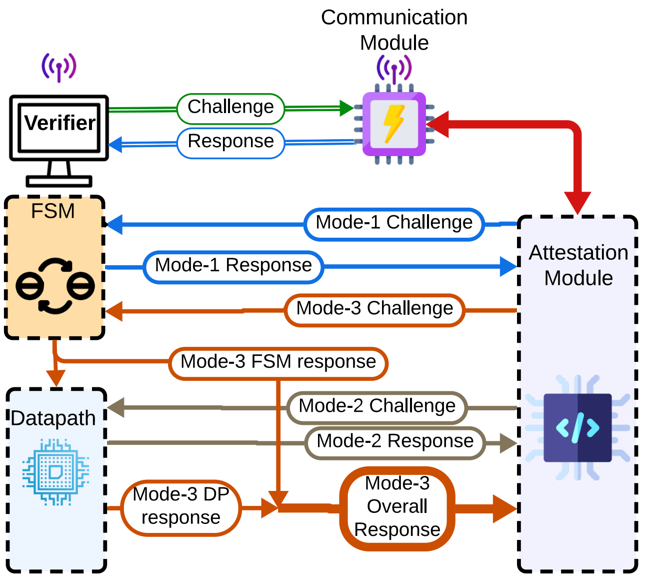

Fig. 4: Attestation Scheme.

1) Attestation of the FSM (Mode-1): The FSM controls the datapath by organizing its operation in various states. It monitors the status of the operation in each state, issues the control signal, and executes the transition between the states. Any unauthorized modifications to the logic of the FSM would result in changes in the control signals and/or the transitions between the states. The proposed scheme attests the FSM by validating the control signals and the transitions between the states. The challenge is designed to offer further flexibility to the *Ve* by allowing partial as well as full attestation. Partial attestation validates a specific portion of the FSM, whereas full attestation validates the entire FSM. The challenge created by the *Ve* consists of the following two elements:

a. **The starting state for attestation.** The starting state for attestation enables the *Ve* to initiate the attestation process from any random state of the FSM. This randomness is crucial for protecting against replay attacks. In our framework, the FSM states are typically represented using a one-hot encoding scheme, where each bit represents a distinct state. Suppose the FSM has four states, i.e., S0, S1, S2, and S3 which can be encoded as ‘0001’, ‘0010’, ‘0100’, and ‘1000’ respectively. The verifier can choose to start the attestation from any state by sending the corresponding one-hot encoded value as the initial state.
b. **The sequence of status signals.** An FSM’s control signals and transition between states are dependent on the received status signals. This enables the *Ve* to validate the FSM’s behavior with the received status signals.

The attestation module located in the *SecPart* then executes this challenge on the *OpenPart* of the *Pr*. Subsequently, the response to this challenge is given by:

a. **The sequence of FSM states.** This part of the response represents the transitions that the FSM undergoes in response to the received challenge. Any unauthorized modification to the FSM's transition logic would result in an unexpected sequence of states.
b. **The sequence of FSM output control signals.** This part of the response represents the control signals that the FSM issues in response to the received challenge. Any unauthorized modification to the FSM's control logic would result in an unexpected sequence of control signals.

The structures of both the challenge and the response in this mode of attestation are shown in Figure 5.

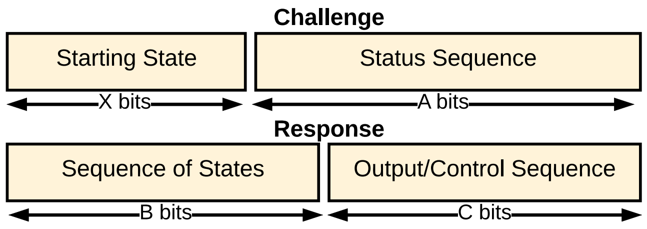

Fig. 5: Challenge-response pair for Mode-1.

2) Attestation of the datapath (Mode-2): The design's datapath executes a particular functionality during a specific state and returns a status signal to the FSM. Unauthorized modifications to the datapath would lead to changes in both the status signals and/or the response time. The proposed scheme leverages the validation of the status signals and response time for attestation of the datapath. Similar to Mode-1, the *Ve* is empowered to conduct partial as well as complete attestation. The *Ve* also has the flexibility of conducting the attestation of any state. The elements of the challenge that the *Ve* generates for this mode are as follows:

a. **The starting state for attestation.** Similar to Mode-1, the starting state enables the *Ve* to validate the behavior of the datapath in any given state.
b. **The sequence of control signals.** The datapath performs different tasks based on the control signals it receives. This part of the challenge enables the *Ve* to validate the datapath's behavior in response to the received control signals.

The attestation module in the *SecPart* executes the received challenge on the datapath and computes its response to the challenge. The response to this attestation also consists of:

a. **The sequence of status signals.** This part of the response represents the task it has performed in response to the given challenge. An unexpected sequence of status signals in response to a given challenge indicates an unauthorized modification to the datapath logic.
b. **The sequence of response time.** This part of the response represents the execution time of a task. Since functions in digital design are strictly tied to the clock, each task has a predetermined execution time. An unexpected sequence of response times indicates an unauthorized modification to the datapath logic.

The response is then relayed back to the *Ve* for comparison with the anticipated response. The structure of the challenge-response pair for attestation of the datapath is depicted in Figure 6.

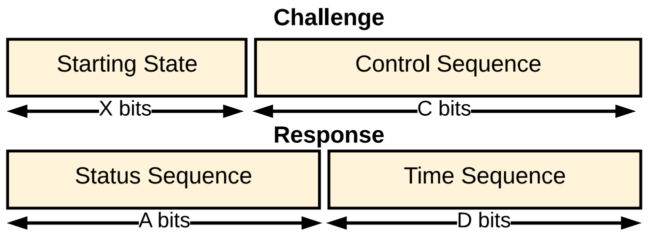

Fig. 6: Challenge-response pair for Mode-2.

3) Attestation of overall design (Mode-3): This mode of attestation is designed to validate the overall behavior of the design by combining the individual attestation of the FSM and the datapath, i.e., mode 1 and mode 2. The structure of the challenge is similar to that of Mode-1, while the response consists of the response of both the FSM and the datapath, as shown in Figure 7.

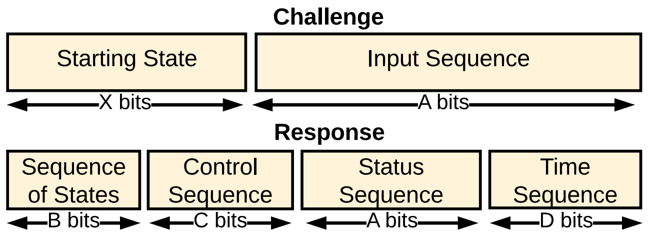

Fig. 7: Challenge-response pair for Mode-3.

The number of bits required to encode challenges and responses for an N-state FSM with the longest path L, W_{cs}-bit control signal, and W_{ss}-bit status signal are given as:

- Starting state : $X = \log_2(N)$.
- Input/status sequence: $A = L \times W_{ss}$.
- State sequence: $B = L \times X$.
- Control sequence: $C = L \times W_{ss}$.
- Time sequence: $D = L \times \log_2(T)$; where T is Maximum possible time.

C. Attestation Protocol

The proposed attestation protocol has two phases, i.e., the enrollment phase and the execution phase.

1) Enrollment phase: The enrollment phase is exclusively performed during the initial configuration of the FPGA and whenever reconfiguration or hardware design modifications are executed. In this phase, the *Co* takes charge and performs the following tasks:

- Configuration/reconfiguration of the *OpenPart*.

- Modify the attestation module in the *SecPart* according to the updated design.
- Provide the *Ve* with a copy of the updated design. The *Ve* utilize this copy to create challenges and expected responses.

Once the enrollment phase is complete, all three components, namely the *Co*, the *Ve*, and the *Pr* are synchronized and hold the updated structure of the hardware design. It is important to note that the enrollment phase enables the easy integration of any permitted upgrades or changes to the design for different targets. This phase provides a mechanism for introducing approved changes to preserve alignment between the attestation scheme and the configurations of the digital design.

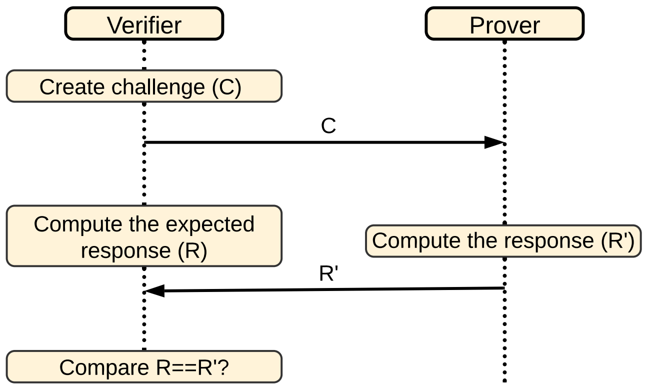

Fig. 8: Attestation protocol.

2) Execution phase: During the execution phase, the *Ve* takes the initiative to start attestation by creating a random challenge. This challenge comprises of an arbitrary starting state and a sequence of random inputs to the component being attested. The use of random input sequence enables the *Ve* to attest any specific portion of the hardware design. Additionally, randomness also enhances the security of the attestation process by making it more robust against replay attacks. To start attestation, the *Ve* generates the challenge and sends it to the *Pr*. Simultaneously, the *Ve* computes the expected response to the sent challenge and retains it as a reference for subsequent comparison. Within the *Pr*, the *SecPart* receives the challenge and commences the attestation process. The attestation module in the *SecPart* then ensures that the target design assumes the specified starting state as per the challenge. Subsequently, it sequentially applies the challenge signal, recording the response of the hardware design for each input. The *SecPart* sends the response back to the *Ve*. The *Ve* compares the received actual response with the stored reference. If any malicious modification has occurred, the comparison would reveal discrepancies, prompting the *Ve* to raise an alarm or take appropriate actions. Figure 8 represents the execution phase of the attestation protocol.

D. Uninterrupted Attestation

We propose to leverage the inherent reconfigurable and redundant characteristics of FPGAs to design an uninterrupted run-time attestation scheme. This exploitation involves employing a redundant replica of the primary design during the attestation process. In FPGAs, input gating logic is used to selectively activate and deactivate the specific components of the design [21]–[23]. Our proposed technique utilizes the original design and a replica. During normal operation, the original design actively executes the intended tasks, while the replica remains inactive. To facilitate attestation without disrupting normal operation, the replica is configured and activated using input gating logic before starting the attestation process. It is necessary to synchronize the original design with the replica before initiating an attestation process. All the contents of the register, memory, and other relevant variables from the original design are copied to the replica using the intermediate buffers. The replica utilizes this information to synchronize with the original design and resume operation after run-time switching. The switching process is shown in Figure 9. Upon successful configuration and synchronization of the replica, the system seamlessly transitions from the normal mode of operation, (with only the original design activated) to attestation mode (with both the original design and the replica design activated). In attestation mode, the *Ve* conducts the attestation of the replica design and compares its behavior to the original design for validation. This comparison enables the attestation of the original design without imposing a halt on normal operations. Upon successful attestation, the system reverts to the normal mode of operation by deactivating the replica design. This dynamic mechanism ensures the uninterrupted operation of the FPGA-based system.

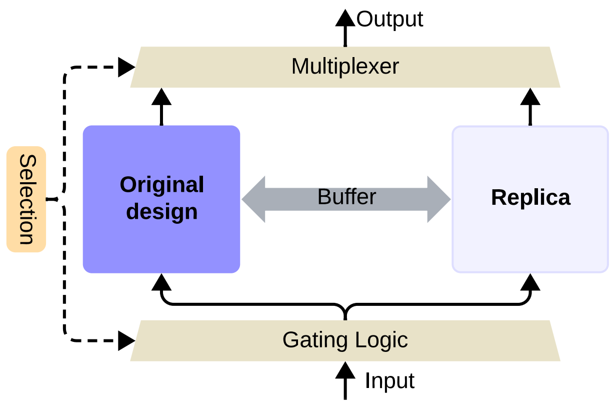

Fig. 9: Switching between the original design and its replica.

V. Security Analysis

This section provides the formal security analysis of the proposed attestation scheme.

The proposed attestation scheme is based on periodic attestation of the *Pr*. The periodic attestation of the *Pr* helps in improving security by reducing the time window for injection and execution of attacks while reducing the resources required to continuously monitor the *Pr*. However, the frequency of attestation plays a crucial role and requires careful consideration. We denote the minimum time required to inject an attack as T_{min}, the time available for execution of this injected attack T depends upon the attestation frequency F. The attestation frequency required to promptly detect the injected modification without being executed, F_{max}, is mathematically expressed as $F_{max} = 1/T_{min}$. Although adopting a higher attestation frequency is useful in reducing the time window available to attackers, it leads to higher resource consumption. Lowering the attestation frequency is a viable strategy to reduce resource requirements, but it comes at the risk of giving potential attackers more time to execute an injected attack, as depicted in Figure 10. Thus, determining an optimal attestation frequency becomes critical and requires thoughtful considerations to strike a balance between effective resource management and defense against potential threats.

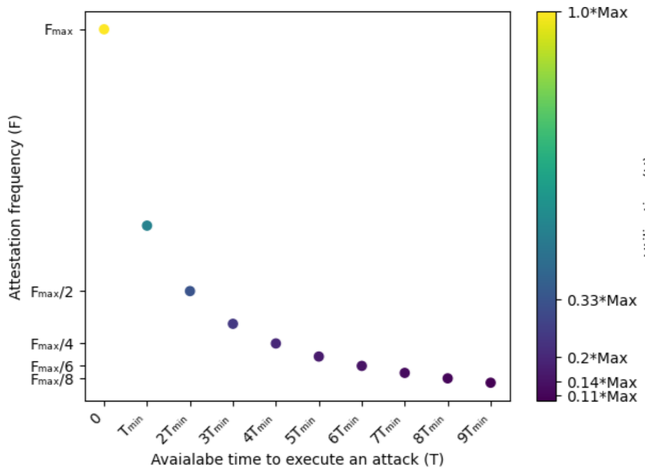

Fig. 10: Tradeoff between the resource utilization and the time available to execute an attack.

Lemma 1: To detect a modification attack with 100% certainty before it is executed with minimal resource utilization, the optimal attestation frequency, F_{optimal}, is given by the following equation:

$$F_{\text{optimal}} = \frac{1}{T_{\text{inject}} + T_{\text{execution}}} \quad (2)$$

Proof: Let the time required for injection of a modification be T_{inject} and the time required for its execution be $T_{\text{execution}}$. If $T_{\text{available}}$ represents the time available for an attack between consecutive attestation iterations, then the probability of an attack being successful is given by:

$$P_{\text{attack}} = \min\left(\max\left(\frac{T_{\text{available}}}{T_{\text{inject}} + T_{\text{execution}}} - 1, 0\right), 1\right) \quad (3)$$

where $0 \leq T_{\text{available}} \leq \frac{1}{F_{\text{attestation}}}$. The probability of successfully detecting an attack can be expressed as:

$$P_{\text{detection}} = 1 - P_{\text{attack}}$$

$$P_{\text{detection}} = 1 - \min\left(\max\left(\frac{T_{\text{available}}}{T_{\text{inject}} + T_{\text{execution}}} - 1, 0\right), 1\right) \quad (4)$$

From Equations (2), (3), and (4) it can be deduced that at optimal point with $F_{\text{attestaion}} = F_{\text{optimal}}$ we get $P_{\text{attack}} = 0$ and $P_{\text{detection}} = 1$ with optimal resource utilzation. An attestation frequency exceeding F_{optimal} leads to increased resource utilization without enhancing detection capabilities. Conversely, reducing the attestation frequency below F_{optimal} diminishes the probability of detection while lowering resource utilization. Therefore, setting the attestation frequency as per equation (2) ensures the maximum detection capability with optimal resource utilization. This configuration guarantees that an attack can be successfully detected without demanding extensive computational resources.

Theorem 1: A successful attestation of an FPGA-based IoT device would ensure that the design is not modified maliciously.

Proof: The adversary *(Ad)* may attempt to launch a modification attack to disrupt the operation of hardware design. The following game between a challenger *(Ch)* and an adversary *(Ad)* is used to model this attack.

1) The *Ch* initiates the proposed attestation protocol between an FPGA-based IoT device and the *Ve*.
2) The *Ad* attempts to introduce a malicious modification in the FSM or its datapath to disrupt the intended operation of the design.
3) The *Ve*, aided by the secure partition (SecPart) of the *Pr*, conducts an attestation process to validate the design.
4) The *Ad* wins the game if it can compromise the integrity of the design undetected at any point during the game.

The *Ad* can only successfully execute the modification attack if it can avoid detection during attestation. We can model the adversary's advantage for successfully executing a modification attack as $\alpha_{Ad}^{Mod1} = P_{\text{attack}}$. As per Lemma 1, an attestation process with $F_{\text{attestaion}} \approx F_{\text{optimal}}$ would detect any modification in the design that disrupts the operation of the design with $P_{\text{detection}} \approx 1$ making $P_{\text{attack}} \approx 0$. Thus, $\alpha_{Ad}^{Mod1} \approx 0$

Theorem 2: A successful execution of the proposed attestation for an FPGA-based IoT device would verify the integrity of both the original design and its replica, ensuring neither has been compromised.

Proof: The *Ad* may attempt to compromise either the original design or the replica design by introducing malicious modifications. This attack is modeled using the following game between *Ch* and *Ad*.

1) The *Ch* initiates the proposed attestation protocol between an FPGA-based IoT device and the *Ve*.
2) The *Ad* attempts to introduce a malicious modification in the original design and/or in the replica design.
3) The *Ve* performs the attestation of the FPGA-based IoT device.
4) The *Ad* wins the game if it successfully introduces a modification in either of the designs that goes undetected during attestation.

The *Ad* can only be successful in introducing a modification to any of the original or replica designs if the modification is not detected during attestation. We can model the adversary's advantage for successfully executing a modification attack at any of the designs as $\alpha_{Ad}^{Mod2} = \max(P_{\text{attack-orignal}}, P_{\text{attack-replica}})$.

As per Lemma 1, any modification in the replica design would be detected during attestation with the probability of successful attack $P_{\text{attack-replica}} \approx 0$. As the original design is compared against the replica design during attestation, a modification in the original design would also be detected with the same probability, i.e., $P_{\text{attack-original}} \approx 0$. thus $\alpha_{Ad}^{Mod2} = \max(0,0) \implies \alpha_{Ad}^{Mod2} \approx 0$. So, by attesting the replica design and comparing it with the original design, the proposed scheme ensures that any modification attack on either design would be detected.

Lemma 2: The proposed protocol is secure against replay attacks.

Proof: The *Ad* may attempt to launch a replay attack by storing the challenge-response pairs to get insight into the hardware design. Since the proposed technique enables the *Ve* to generate a distinct challenge for each attestation attempt by creating a unique sequence of inputs and starting from a random state, the system prevents replay attacks. The *Ad* cannot reuse previous challenge-response pairs because each attestation involves a new, unpredictable challenge. This ensures the integrity and security of the attestation process by rendering replay attacks ineffective.

VI. Attestation Example

We use the following example to provide a detailed illustration of the attestation process. In this example, we consider an attack similar to attacks considered in [24], [25] to exploit the BRAM of an FPGA device. The adversary in this scenario aims to corrupt the critical AES data being stored in BRAM by manipulating the control sequence of an FSM that controls the storing process. The FSM for this design encompasses seven states, as shown in Figure 11. During attestation, the attestation module receives a challenge from the *Ve* and executes the challenge on this design to validate its behavior. The control signal corresponding to the store state, originally denoted as '100', undergoes a malicious alteration by the attacker to '001'. During the attestation process, a response to any challenge to the FSM that traverses through the store state may yield a discrepancy between the expected and actual outcomes. To show this, we consider a specific challenge where the *Ve* conducts the attestation of the FSM by initiating a sequence that starts in a waiting state and ends in the same state, going through all the intermediate states in a closed loop. Table II shows the challenge's input sequence and the subsequent expected and actual FSM responses. A discernible discrepancy manifests between the expected and actual control sequences, detecting the malicious modification introduced during the attestation process.

VII. Experiments

A high-level implementation of the attestation scheme on an FPGA is shown in Figure 12. The intended design contains the actual hardware design to be attested. The attestation module residing in the secure part of the *Pr* plays a central role in the attestation process and is granted access to both the FSM and the datapath. It receives the challenge as well as the other attestation information from the *Ve*. The attestation module, after setting up the attestation phase, executes the elements of the challenge iteratively and stores the corresponding element of response in the response sequence. We utilize this architecture to conduct the attestation of the design under test for simulation as well as implementation purposes.

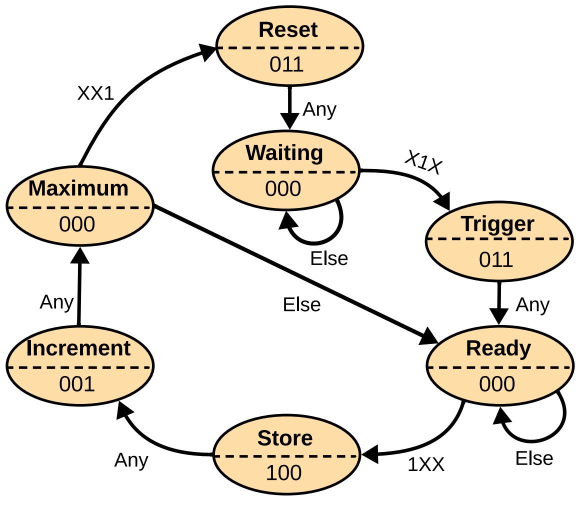

Fig. 11: FSM for attestation example.

TABLE II: Comparison of the Challenge, Expected Response, and Actual Response.

Challenge	Expected Response		Actual Response	
	State Sequence	Control Sequence	State Sequence	Control Sequence
010	Trigger	011	Trigger	011
010	Ready	000	Ready	000
100	Store	**100**	Store	**001**
100	Increment	001	Increment	001
100	Maximum	000	Maximum	000
001	Reset	011	Reset	011
001	Waiting	000	Waiting	000

We leveraged the hardware design of a basic oscilloscope to simulate the proposed attestation scheme. Various modifications to the original hardware were simulated to validate the efficacy of the proposed attestation scheme. The Xilinx Vivado 2019.1 suite was used to simulate and implement the proposed attestation scheme, accordingly the *Ve* and attestation module were developed in VHDL. The list of various separately simulated modifications to the original design is given as follows:

i. Insertion of an additional state.
ii. Modification of control signals in a state.
iii. Modification of status signals.
iv. Addition of a transition, i.e., from Waiting state directly to Store state.
v. Insertion of logic in the datapath.

Each modified design was simulated with the attestation module by issuing random challenges targeting the affected portions. The responses to these challenges were compared against the respective expected responses computed by the *Ve*. Table III shows the detection of various modifications with

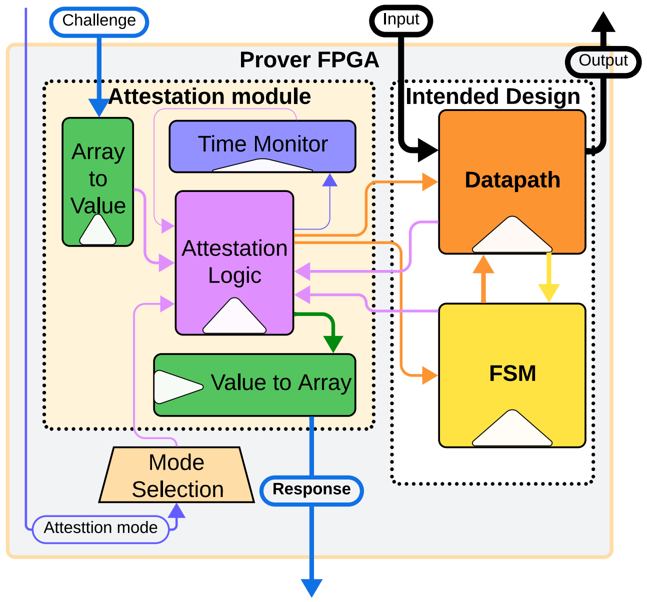

Fig. 12: A high-level implementation of the attestation scheme.

respective detection modes and response sequences. Every modification made to the original design was detected successfully by the proposed attestation scheme. This shows that the proposed attestation technique is capable of detecting unauthorized or malicious modifications to the hardware design.

TABLE III: Simulation results.

Modification No.	Detection Modes	Detection Response
i.	Mode-1 & Mode-3	State sequence
ii.	Mode-1 & Mode-3	Control sequence
iii.	Mode-2 & Mode-3	Status sequence
iv.	Mode-1 & Mode-3	State sequence & Control sequence
v.	Mode-2 & Mode-3	Time sequence

To demonstrate the resource efficiency of the proposed attestation scheme, we implemented the prototypes of two different designs on the ZedBoard from Digilent. This board features a Xilinx Zynq-7000 AP SoC XC7Z020-CLG484, containing 7-Series programmable logic, Dual-core ARM Cortex-A9 processor, 512 MB DDR3 memory, and 256 MB Quad-SPI Flash. The primary objective of this implementation was to assess the resource utilization required for the attestation process. Considering that the *Ve* is not resource-limited, we did not conduct a resource utilization analysis for it. Therefore, this study concentrates on the resource utilization analysis for implementation of the *Pr* on an FPGA.

Prototype 1. This prototype implements the same hardware design utilized for simulation. Table IV shows the resource utilization of the proposed technique for prototype 1 on the target FPGA. Various resources considered include Look-Up Tables (LUTs), Flip Flops (FFs), Block Random Access Memory (BRAM), Mixed-Mode Clock Managers (MMCMs), and power consumption of the design.

TABLE IV: Resource utilization of the prototype 1.

	LUTs	BRAM	FFs	MMCM	Power (mW)
Available	53200	140	106400	4	—
Intended Design	583	1	485	2	405
Attestation module	39	0	26	0	78
Attestation module to system design percentage	6.68	0	5.36	0	19.26

TABLE V: Resource utilization of the prototype 2.

	LUTs	BRAM	FFs	MMCM	Power (mW)
Available	53200	140	106400	4	—
Intended Design	2169	0	551	1	220
Attestation module	124	0	41	0	65
Attestation module to system design percentage	5.72	0	9.09	0	29.54

Prototype 2. This prototype implements the hardware design of a Multi-Layer Perceptron (MLP) classifier. The MLP classifier has four inputs, two hidden layers, and three outputs. The first hidden layer is comprised of 10 neurons, whereas the second layer has 8 neurons. The FSM of the design has 16 states to control one complete cycle of the network. Resource utilization for prototype 2 is illustrated in Table V.

TABLE VI: Comparison of the proposed technique with existing attestation techniques in terms of resource overhead.

	[1]	[26]	[8]	Prototype 1	Prototype 2
BRAM overhead	9.6%	12%	1.47%	0%	0%
CLB overhead	8.9%	26%	8%	6.02%	7.42%

Table VI compares the resource overheard of existing techniques to the two implemented prototypes of the proposed technique. The proposed technique offers a notable resource efficiency compared to existing attestation methods, as highlighted in Table VI. The proposed scheme stands out when analyzing the utilization of BRAM because it incurs no additional cost (0%) in both the prototypes, in contrast to competing methods [1], [26], and [8], which have BRAM overheads of 9.6%, 12%, and 1.47%, respectively. Similarly, the proposed strategy retains a minimal utilization of CLB resources with an average overhead of only 6.02% for prototype 1 and 7.42% for prototype 2 as opposed to the larger overheads of 8.9%, 26%, and 8% incurred by the techniques proposed in [1], [26], and [8], respectively.

Table VII provides a breakdown of the attestation latency, focusing on the number of clock cycles, frequency of execution, and the time taken for various actions within the process. It's important to note that the table does not include the time required for communication and related processing, as these aspects are typically dependent on external factors such as network latency and system overhead. The total time required to generate a response to a challenge of length L_{seq} is the sum of the individual times and is given as $0.04 + 0.03 \times L_{seq}$.

TABLE VII: Attestation latency.

Action	**Clocks**	**Repeats**	**Time**
Challenge receiving	1	L_{seq}	$0.01 \times L_{seq}$ us
Mode selection	1	1	0.01 us
Initialization	3	1	0.03us
Attestation	1	L_{seq}	$0.01 \times L_{seq}$ us
Response sending	1	L_{seq}	$0.01 \times L_{seq}$ us
Total			$0.04 + 0.03 \times L_{seq}$ us

VIII. Conclusion

In this paper, a novel lightweight attestation technique was proposed to verify the hardware design deployed into an FPGA. The proposed scheme enables remote run-time self-attestation of FPGA devices and is based on using the challenge-response mechanism to detect any unauthorized modification in the hardware structure of a design. The proposed technique provides an alternate way to the conventional techniques employing complicated cryptographic functions to detect modification in FPGA bitstreams. The effectiveness of the proposed attestation scheme was validated through simulations, which successfully identified various unauthorized modifications introduced into the original hardware design. The implementation results of the proposed technique on a Zynq-7000 SoC board show that the proposed technique is efficient in terms of used resources and outperforms the existing attestation techniques.

While the effectiveness of the proposed attestation mechanism in detecting unauthorized hardware modifications is evident, there are specific avenues for further enhancing this research:

- Identification of data-stealing trojans that operate covertly without causing operational disruptions.
- Assess the viability of utilizing machine learning for creating intelligent challenge sequences that optimize the detection of trojans.
- Create a resilient and reliable protocol for intelligently scheduling the attestation process to meet specific security goals.

References

[1] J. Vliegen, M. M. Rabbani, M. Conti, and N. Mentens, "Sacha: Self-attestation of configurable hardware," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 746–751, 2019.

[2] T. Raikovich and B. Fehér, "Application of partial reconfiguration of fpgas in image processing," *6th Conference on Ph.D. Research in Microelectronics & Electronics*, pp. 1–4, 2010.

[3] H. Li, Q. Liu, and J. Zhang, "A survey of hardware trojan threat and defense," *Integr.*, vol. 55, pp. 426–437, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:12801225

[4] M. N. Aman, M. H. Basheer, S. Dash, J. W. Wong, J. Xu, H. Lim, and B. K. Sikdar, "Hatt: Hybrid remote attestation for the internet of things with high availability," *IEEE Internet of Things Journal*, vol. 7, pp. 7220–7233, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:216322415

[5] S. Mal-Sarkar, A. R. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in fpga devices: threat analysis and effective counter measures," in *ACM Great Lakes Symposium on VLSI*, 2014.

[6] S. Bhasin, J.-L. Danger, S. Guilley, X. T. Ngo, and L. Sauvage, "Hardware trojan horses in cryptographic ip cores," *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 15–29, 2013.

[7] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," *2010 IEEE Symposium on Security and Privacy*, pp. 159–172, 2010.

[8] R. Chaves, G. K. Kuzmanov, and L. Sousa, "On-the-fly attestation of reconfigurable hardware," *2008 International Conference on Field Programmable Logic and Applications*, pp. 71–76, 2008.

[9] C. Basile, S. D. Carlo, and A. Scionti, "Fpga-based remote-code integrity verification of programs in distributed embedded systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 187–200, 2012.

[10] P. Kumar and R. Srinivasan, "Detection of hardware trojan in sea using path delay," *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pp. 1–6, 2014.

[11] F. Devic, L. Torres, and B. Badrignans, "Secure protocol implementation for remote bitstream update preventing replay attacks on fpga," *2010 International Conference on Field Programmable Logic and Applications*, pp. 179–182, 2010.

[12] B. Badrignans, R. Elbaz, and L. Torres, "Secure fpga configuration architecture preventing system downgrade," *2008 International Conference on Field Programmable Logic and Applications*, pp. 317–322, 2008.

[13] M. Bach, A. Werner, M. Mrozik, and K. A. Cyran, "A hierarchy of finite state machines as a scenario player in interactive training of pilots in flight simulators," *International Journal of Applied Mathematics and Computer Science*, vol. 31, pp. 713 – 727, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:245828440

[14] "AMD Adaptive Computing Documentation Portal." [Online]. Available: https://docs.xilinx.com/v/u/en-US/xapp1159-partial-reconfig-hw-accelerator-zynq-7000

[15] Y. Xue, P. Cronin, C. Yang, and J. Hu, "Non-volatile memories in fpgas: Exploiting logic similarity to accelerate reconfiguration and increase programming cycles," in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2015, pp. 92–97.

[16] F. J. Streit, F. Fritz, A. Becher, S. Wildermann, S. Werner, M. Schmidt-Korth, M. Pschyklenk, and J. Teich, "Secure boot from non-volatile memory for programmable soc architectures," *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 102–110, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:215828348

[17] K. W. Gear, A. Sánchez-Macián, and J. A. Maestro, "An analysis of fpga configuration memory seu accumulation and a preventative scrubbing technique," *Microprocess. Microsystems*, vol. 90, p. 104467, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246744115

[18] D. E. Owen, D. Heeger, C. Chan, W. Che, F. Saqib, M. Areno, and J. F. Plusquellic, "An autonomous, self-authenticating, and self-contained secure boot process for field-programmable gate arrays," *Cryptogr.*, vol. 2, p. 15, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52972331

[19] G. Pocklassery, W. Che, F. Saqib, M. Areno, and J. F. Plusquellic, "Self-authenticating secure boot for fpgas," *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 221–226, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49193478

[20] S. Trimberger and J. Moore, "Fpga security: Motivations, features, and applications," *Proceedings of the IEEE*, vol. 102, pp. 1248–1265, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:207022570

[21] Z. Zhang, L. L. Njilla, C. A. Kamhoua, and Q. Yu, "Thwarting security threats from malicious fpga tools with novel fpga-oriented moving target defense," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, pp. 665–678, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:67870623

[22] M. Hosseinabady and J. L. Núñez-Yáñez, "Run-time power gating in hybrid arm-fpga devices," *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–6, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:16700087

[23] A. A. M. Bsoul and S. J. E. Wilton, "An fpga architecture supporting dynamically controlled power gating," *2010 International Conference on Field-Programmable Technology*, pp. 1–8, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:260662378

[24] Y. Zhang, F. Zhang, B. Yang, G. Xu, B. Shao, X. Zhao, and K. Ren, "Persistent fault injection in fpga via bram modification," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, 2019, pp. 1–6.

[25] D. Gnad, "Remote attacks on fpga hardware," Ph.D. dissertation, Dissertation, Karlsruhe, Karlsruher Institut für Technologie (KIT), 2020, 2020.

[26] K. Xia, Y. Luo, X. Xu, and S.-H. Wei, "Sgx-fpga: Trusted execution environment for cpu-fpga heterogeneous architecture," *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 301–306, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:243893350

Muhammad Usama is a Ph.D. candidate in the Computer Science and Engineering Department at the University of Nebraska, Lincoln, USA. He received his bachelor's degree in Electrical Engineering from the National University of Computer and Emerging Sciences, Pakistan, in 2015, followed by a master's degree in Electrical Engineering from the Lahore University of Management Sciences, Lahore, Pakistan, in 2018. His research interests include the security of critical infrastructures, hardware security for reconfigurable devices, and machine learning applications for improving cyber-physical security.

Muhammad Naveed Aman is an Assistant Professor at the University of Nebraska-Lincoln (UNL). Previously, he served as an Assistant Professor with the Faculty of the National University of Computer and Emerging Sciences, Pakistan. He also served as a Senior Research Fellow with the School of Computing, National University of Singapore, Singapore. He received the B.Sc. degree in computer systems engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2006, the M.Sc. degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, in 2008, and the M.Eng. degree in industrial and management engineering and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2012. He heads the Goof-proof Hardware Oriented Security and Trust (GHOST) lab at UNL. His extensive research spans hardware systems security in embedded devices, physical layer security for IoT devices, and trustworthy machine learning. Noteworthy achievements include pioneering device attestation algorithms, innovative approaches to physical layer security leveraging transceiver and wireless channel characteristics, and contributions to understanding privacy attacks on machine learning models. Dr. Aman's interdisciplinary expertise extends to blockchains, power systems, optimization, and control systems.

Biplab Sikdar received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was on the faculty of Rensselaer Polytechnic Institute from 2001 to 2013, first as an Assistant and then as an Associate Professor. He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include wireless network, and security for IoT and cyber physical systems.