Journal of Hardware and Systems Security (2021) 5:260–268
https://doi.org/10.1007/s41635-021-00123-3

A Secure Boot Framework with Multi-security Features and Logic-Locking Applications for Reconfigurable Logic

Geraldine Shirley Nicholas[1] · Ali Shuja Siddiqui[1] · Sam Reji Joseph[1] · Gregory Williams[1] · Fareena Saqib[1]

Received: 1 March 2021 / Accepted: 4 November 2021 / Published online: 29 November 2021

Abstract
Reconfigurable platforms such as field-programmable gate arrays (FPGAs) are widely used as an optimized platform with fast design time. New features such as dynamic reconfiguration make the bitstream vulnerable to clone/modification attacks which raise a security concern in today's heterogeneous computing architecture. A widely adopted countermeasure is by providing a secure boot mechanism as root-of-trust to authenticate the unmodified firmware to prevent attackers from manipulating it. In this work, we propose an automated security-aware design flow scheme by integrating the logic-locking scheme for secure boot in Xilinx FPGAs. The proposed design implements FPGA-based logic obfuscation, with a pre-boot in-field device authentication scheme implemented using ARM TrustZone enabled with Trusted Platform Modules (TPM) key provisioning. This scheme constructs security features that can protect the IPs during the design process and integrates the primitives with FPGAs secure boot process and enhances bitstream security.

Keywords Hardware Security · FPGA · Secure boot · ARM trustZone · Logic obfuscation · Bitstream security · TPM

1 Introduction

The semiconductor industry faces a major issue with the rise of counterfeit ICs in the global supply chain. This horizontal semiconductor supply chain model also invites miscreants with malicious intent to enter the supply chain [1]. The attacker can tamper and insert hardware trojans which poses a threat to the integrity of information stored in the ICs and FPGAs. FPGAs are vulnerable to attacks such as reverse engineering, bitstream cloning, and IP theft.

To mitigate these attacks, techniques such as IC camouflaging [2], split manufacturing [3], and logic locking [4] have been proposed for the ASIC and FPGAs. Reconfigurability and on-the-fly updates in FPGAs open a medium for attackers to have physical access to the device by modifying the bitstream. One way to secure the bitstream is by implementing a secure boot process [5] with authentication which provides root of trust, but this can be circumvented by tampering with the device boot process as the bitstream can be modified at runtime using processor configuration access port (PCAP) or internal configuration access port (ICAP) port. The programmable logic (PL) can be replaced to perform an entirely different operation by replacing it with a malicious bitstream during boot or runtime. Cryptographic processors such as Trusted Platform Modules (TPM) [6] are used along with secure boot for key provisioning where the keys for authentication can be stored in nonvolatile memory to mitigate the attacker access to the keys via invasive attacks.

In the heterogeneous SoC design architecture, the ARM-centric processing system provides the hardware-assisted TrustZone feature for secure implementation and the FPGA fabric holds the programmable logic which uses the Advanced Extensible Interface (AXI) bus to establish communication with the processing system [7]. This work focuses on designing the security features by implementing logic locking and integrating them in the design flow by inserting key gates, and its application for secure boot, where the design/logic functions as intended only when the correct key combination is given. The TrustZone provides a secure authentication while booting up the logic-locked bitstream in the secure world. The TPM which is interfaced with the FPGA provides the key and is only accessible through the ARM TrustZone's secure world.

✉ Geraldine Shirley Nicholas
gnichola@uncc.edu

Fareena Saqib
fsaqib@uncc.edu

[1] University of North Carolina at Charlotte, Charlotte, North Carolina, USA

1.1 Contributions

This paper makes the following contributions:

1. A novel mechanism for implementing FPGA-based logic obfuscation that is intended to function correctly only for the correct key combination.
2. A pre-boot in-field device authentication scheme is extended to implement runtime secure boot and bitstream security.
3. A remote client-server authentication scheme is proposed that uses ARM TrustZone to ensure a secure configuration and secure boot mechanism.
4. TPM integration for key provisioning in a secure way by providing driver libraries for security functions are open-sourced.

1.2 Paper Organization

The paper is organized as follows. Section 2 lists the related work for securing the reconfigurable logic with a logic-locking-based design for trust techniques establishing the secure boot process. Section 3 discusses the security vulnerabilities in FPGAs. Section 4 describes the proposed framework, and Sect. 5 presents the experimental setup and results of the proposed framework and security analysis is discussed in Sect. 6.

2 Related Works

2.1 Secure Boot for FPGAs

SoC FPGAs have programmable logic and supports soft-IP-based or hardwired microprocessors [8]. The reconfigurable fabric on FPGAs is programmed using bitstreams which configures the Look Up Tables (LUTs) in the logic fabric. Bitstreams also configure other fabric elements, e.g., on-chip memory, digital system processing (DSP), clocking blocks, and wire connections. An attack on the bitstream can affect the entire system operation of a device on the field.

The attacker can redirect the normal flow of execution to an unauthorized piece of code by hijacking the boot flow [9, 10]. In the reconfigurable computing domain, SRAM-based FPGAs allow modification in the field. The PL is programmed at boot time and this process is either performed by the Zeroth Stage Boot Loader software commonly referred to as BootROM or by the First Stage Boot Loader (FSBL) [11] depending on the type of FPGA configuration. If the FPGA is equipped with a Processing System (PS), it controls the loading of the PL bitstream, otherwise, BootROM takes care of the PL bitstream loading process. Figure 1 shows the FPGA boot process.

Countermeasures such as reconfiguring during runtime by dynamic partial configuration (DPR) and using physical unclonable function (PUF) for authentication and configuring the programmable logic secures the bitstream during secure boot [12]. The responses generated by the PUF acts as an authentication key which in turn allows the boot loader to boot up the application. Secure root of trust architecture with TPM drivers and over-the-air updates to identify malicious modifications in configuration files is implemented in [13].

A multilayered secure boot that updates the LUT frames by modifying the boot image with remote attestation using PUF for mutual authentication during runtime [14]. Another self-authentication secure boot mechanism [15] uses PUF-based authentication in which the secure boot process is protected such that any modification made to unencrypted bitstream results in key regeneration failure of the PUF. To secure open-source architectures a lightweight RISC-V-based secure boot framework with PUF and different encryption standards with secure remote key attestation is implemented [16]. These schemes provide secure boot but are expensive to develop as the storage and communication cost are high. This paper focuses on the security of the bitstream during secure boot to eliminate modification of the bitstream by an adversary. A strong PUF is used to generate the authentication keys for a logic-locked bitstream in a client–server environment to provide mutual trust.

2.2 Trusted Platform Module (TPM)

The Trusted Platform Modules are hardware-based cryptographic processors that provide tamper-resistant non-volatile memory to hold the keys for different encryption functions.

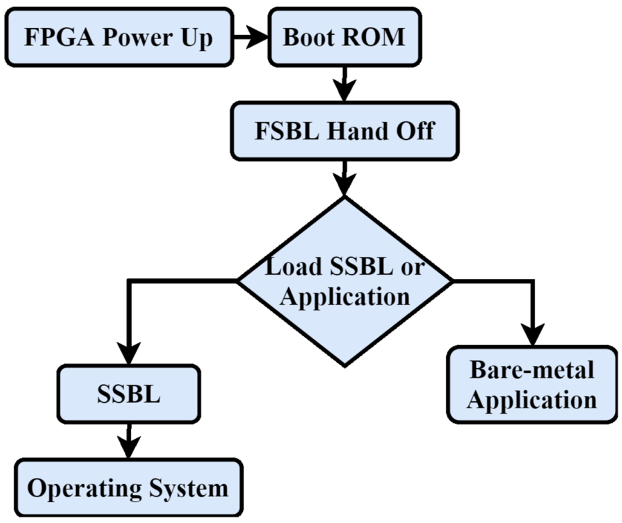


Fig. 1 FPGA Boot Process

It has a true random number generator which along with different special registers called Platform Configuration Registers (PCR) can be configured to provide necessary security functions for a secure boot mechanism [6]. Authentication, key generation, and storage hierarchy for private sensitive applications with over the date updates are different features supported by TPM [17]. TPM integrated with FPGA boot process at the first stage boot loader for client–server key provisioning provides a bitstream verification scheme with secure features [13].

TPM-based key management system secures the recovered key from any leakage by implementing an algorithm to protect the key management module [18]. DFCloud which is a secure data access control mechanism with TPM key encryption management and key sharing for cloud storage services provides security features for data leakages [19]. TPM service functions along with TrustZone isolation provides a secure region and root of trust against different software attacks.

2.3 Logic Obfuscation

Logic obfuscation is a design for trust technique that is used to lock the netlist by inserting key gates to the original design. This technique hides the functionality of the design and only allows the correct key combination to unlock the design functionality. The locked design will produce corrupted outputs if an adversary tries to access or modify the design. Figure 2 shows the modified logic-locked equivalent circuit with key gate insertion for the c17 circuit. An input sequence of 10011 and the corresponding outputs X and Y are 01 where the correct key is (K0 and K1=11). If the key is 00 the output of the circuit is modified and leads to an erroneous output.

Attacks such as the Boolean satisfiability attack eliminates wrong key combinations using the distinguished inputs effectively breaking the logic-locking techniques. The SAT resilient techniques such as SARLock and TTLock makes the attack iterations grow exponentially

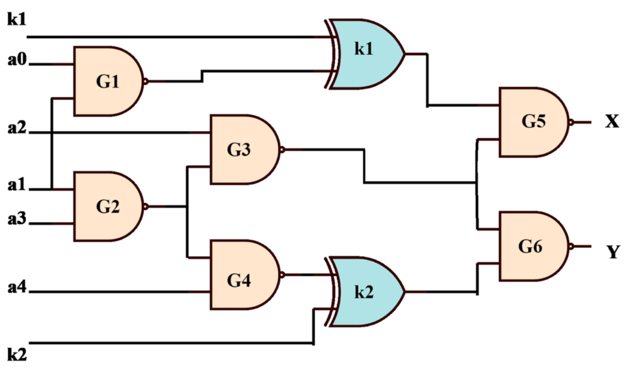


Fig. 2 Logic-locked circuit with two new key gates added in C17 circuit

with increasing key size [20, 21], Anti-SAT that provides tuning flexibility for the key gate configurations [22], SFLL scheme that removes a functional logic block and restores the original logic using Hamming distance [23], and fault-based logic encryption which leverages EDA tools to insert key gates with fault impact metrics [24] are implemented. Logic-locking techniques are used for IP protection that can be extended to applications of secure boot. **We demonstrate an automated application framework implementing SARLock and fault-based insertion schemes that are resilient to SAT attacks and develop its application for FPGA secure boot mechanism.

2.4 ARM TrustZone

The TrustZone technology in ARM provides hardware isolation and prevents software attacks by partitioning it into two worlds, where the secure world protects the critical data, and the non-secure world executes the normal operating system [25]. The secure monitor call acts as a bridge between both worlds. TrustZone enables a Trusted Execution Environment (TEE) where once the system boots, the processor enters the secure world and once all the privileged operations are completed it switches back to the normal world and yields control to the bootloader [26]. System operational modes and device configuration control the data routed to a specific world.

The TEE provides secure trusted services such as authentication and remote attestation to protect the integrity of the application. Some approaches are the TEE enabled authentication from a remote device to mitigate phishing attacks [27]. TrustZone architecture provides run-time authentication and data protection mechanism based on identity authentication to verify the private data and ensuring data access security by making use of the API calls [28]. By using the AXI interconnect signals the PL master dynamically configures the data [29] and a secure authentication scheme can be achieved by using the secure slave key transaction with the master. We propose a secure mechanism in which once the secure boot process is authenticated, the logic-locked bitstream encryption key is sent from the server to the secure IP in the PL and further AES encryption is performed to protect the key from AXI attacks. The master dynamically configures and authenticates the server and decryption is done in a secure environment.

3 Attack/Threat Model

This section overviews the security threats of boot and runtime security of a reconfigurable device in the field. Bitstream spoofing is an attack that updates the FPGA device with an update that may seem to come from an authorized source by using relay and replay attacks [30]. This can gain control over the bitstream after authentication by obtaining the key for bitstream encryption.A bitstream can be modified at runtime using PCAP or ICAP port [31]. An attacker can either replace the PL logic to perform entirely different tasks, add or remove functionality (e.g., hardware trojan), or may even add a leakage side channel for secret information extraction. The points of attack for a reconfigurable device can be at boot or during runtime. At boot, before the bitstream is loaded, an attacker may replace the bitstream with a malicious bitstream. Whereas at runtime, once the bitstream has been loaded an attacker may target dynamic reconfigurable partitions or may want to target certain portions of the configuration.

Malicious code modification can be done during boot time or dynamically during runtime by the insertion of trojans directly on the FPGA configuration bitstream [32]. We focus on bitstream security to mitigate malicious code modification during the boot-up process and runtime. The device power's on and the secure boot authentication is done using the PUF generated challenge-response pairs in a client-server environment. Once the authentication is done the logic-locked bitstream is programmed in the PL secure IP using the ARM TrustZone configurations where the key to unlock the logic-locked bitstream is securely stored in the TPM and through secure communication, it is used to unlock the bitstream.

4 Implementation

We propose a multilayered secure boot mechanism in a client-server model in which authentication is done using PUF, and the correct configuration is unlocked by the logic-locking integrated at the boot process. First the authentication phase is completed, and once the device authenticates the server shares the logic-locked bitstream and loads on the PL's secure IP using ARM TrustZone configurations. The logic-locks key to unlock the bitstream is stored securely in the TPM, that are accessed using drivers as the TPM calls are made before the system is loaded. Once the authentication is completed the logic-locked key is used to successfully unlock the bitstream.

4.1 Secure Boot Mechanism

In the boot process, the first stage acts as a secured encryption/decryption unit providing authentication for the bitstream with the unique key responses generated by the PUF following a second stage in which the logic-locking mechanism is applied to the application logic. A strong PUF is used to generate the per-device unique responses, based on the input challenges. Initially, in the PUF enrollment phase, the challenge-response pairs are generated between the verifier and the prover and to prevent the adversary from gaining access to these response pairs they are encrypted using AES core.

The client is enrolled with the server in a trusted environment and receives the authentication bitstream which is loaded during the boot process. The HELPUF [33], is based on path delay variation, is used to generate the challenge-response pairs for authentication. The server sends the input challenges to the client and the responses are gathered in the server which produces a unique key for authentication. The first stage bootloader loads the authentication bitstream on the PL fabric and the input challenge(c) is given to the PUF from the TPM to generate the response(r).

In the reconstruction phase, the authentication is done by using the unique per device key. Each time the PUF creates a new response for authentication using the unique challenge and response pairs. The PUF response is further processed to generate the secret key for decrypting the encrypted application bitstream. The FSBL overwrites the authentication bitstream that constitutes a PUF and encryption engine with the application bitstream. This acts as a root of trust mechanism establishing secure communication between a client-server model. Figure 3 depicts the overall key exchange mechanism for authentication. TPM is used to securely store the keys generated during runtime and to mitigate malicious intrusion in gaining access to the keys. Custom device drivers are written to enable the device with TPM communication during the secure boot process [13].

4.2 Obfuscation Framework

A logic-locking automation framework and its integration with the secure boot flow in FPGA is implemented, where the key gates can be inserted at the RTL, or netlist level.

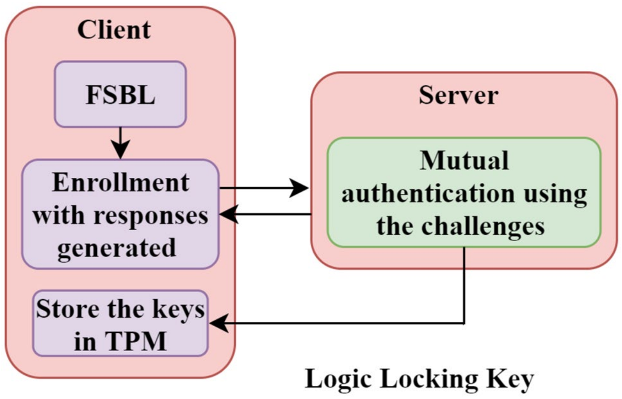

Client
FSBL
Enrollment with responses generated
Store the keys in TPM
Server
Mutual authentication using the challenges
Logic Locking Key

Fig. 3 Secure boot authentication and key exchange process

In this scheme, the key gates are inserted in the gate-level netlist to protect the gate level IP. The key gate insertion scheme is shown in Fig. 4, in which the framework implements and generates the flow integrated with Synopsys tools (Design Compiler) and Vivado. The automated framework secures the IP reuse by integrating key gates in the design flow. Utilizing only the RTL code makes it vulnerable and easier for the adversary to reverse engineer the bitstream [34]. The framework is designed to change the method of implementation upon checking the input file type and the ABC tool [35] is used to convert the input file to Verilog. The test bench generator generates a test bench based on the input HDL and the generated output is a set of vectors to verify the design. During the synthesis process, the RTL is converted into a gate-level netlist using the Synopsys tool. A generalized TCL script is implemented to automate the framework along with the phyton script to insert the logic gates to the netlist.

The experiment includes SARLock and fault-based encryption-based logic locking and maybe further enhanced with the state of art logic-locking schemes resilient to SAT attacks. SARLock increases the key size exponentially to generate computational complexity ensuring that only a single key yields a fault for any input pattern and fault-based encryption integrates fault impact metric [36] to determine the highest fault impact to insert the gates that maximizes the effectiveness of each gate inserted in the design. The key generated during logic locking is stored on the server. If the adversary tries to modify the bitstream, it produces a corrupted or wrong output which makes it unfeasible to clone the IP. During runtime, until the correct key is provided as input from the server, the application's original functionality is unknown and difficult to break. Only after device authentication is successful, the correct key is sent from the server and stored in the device TPM.

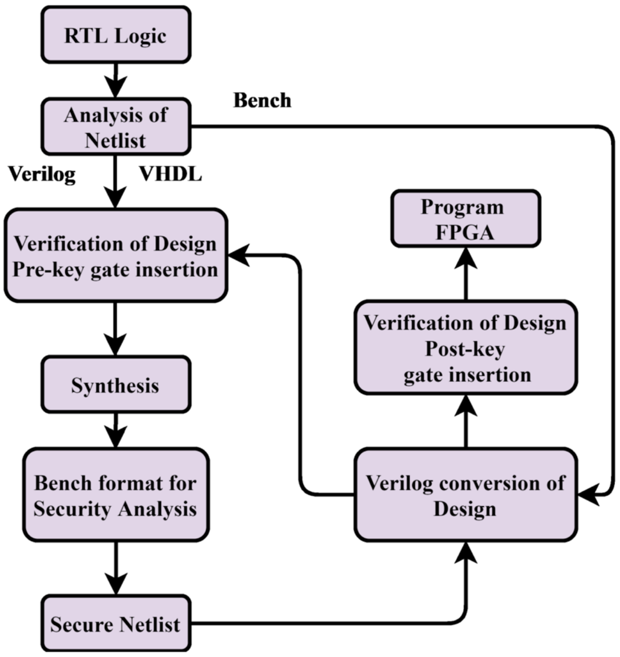

RTL Logic
Analysis of Netlist
Bench
Verilog
VHDL
Verification of Design Pre-key gate insertion
Program FPGA
Verification of Design Post-key gate insertion
Synthesis
Bench format for Security Analysis
Verilog conversion of Design
Secure Netlist

Fig. 4 Key gate insertion flow

4.3 Secure IP Using ARM TrustZone

The ARM TrustZone configuration provides accessibility of MIO ports through the secure world. TPM is integrated with the FPGA using the SPI interface and key storage is done using the TPM driver library. The transfer functions are implemented to establish secure communication at the FSBL to perform the secure boot [13]. Register configuration by PS allows it to design a secure IP in the PL using AXI interconnects. The IP security status is a parameter provided by the AXI interconnect.

Once the secure boot authentication is done the ARM TrustZone is used to load the logic-locked bitstream on the secure IP of the PL. If a non-secure master tries to access the secure IP an error signal is raised by the secure IP. The NS bits on the AXI bus is used for security transaction status. The AXI bus consists of five communication signals as illustrated in Fig. 5, to establish communication between a master and the slave. The ARPROT and AWPROT are the two signals which are used for read/write access of secure and non-secure IPs. Depending on the NS bits the slave checks whether there is a security violation and the transaction is completed with a read or write signal. A core wrapper is implemented to monitor the AXI-transactions of the master–slave interface. The wrapper ensures authorized transactions and modification to the configuration raises an exception. Thus, the secure transaction is done in the secure IP of the PL using ARM TrustZone. The key to unlock the bitstream which is stored in the TPM is securely sent to the secure IP after device authentication.

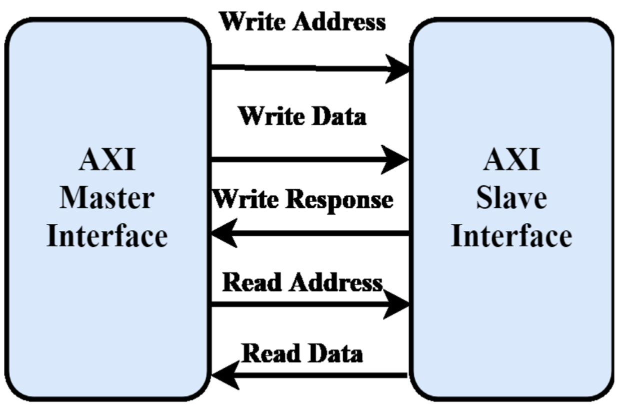


Fig. 5 AXI communication signals

5 Experimental Results

The proposed framework has been implemented on Xilinx Zedboard FPGA which is equipped with Zynq-7000 XC7Z020-CLG484. The ARM Cortex A9 processor is embedded with the FPGA and Fig. 6 shows the hardware setup integrated with the TPM SLB9670 module. The HELPPUF component which is used for generating unique key pairs is integrated into the existing hardware functions [33]. The PUF generates the device-unique encryption key for the authentication bitstream. This encryption key which is 128-bits is used by the AES cryptographic core to encrypt the bitstream and valid verification is completed. Figure 7 shows the system block with PUF IP and the AES IP added as secure slave registers with a custom configured system wrapper. Each GPIO port is 32-bit wide, and the PL is programmed with the authentication bitstream and verified with the PUF generated keys. Once device authentication is done the logic-locked application bitstream is sent from the server to the device.

The obfuscated automated framework with SARLock and fault-based encryption consists of outputs generated by the unobfuscated circuit, the outputs generated by the obfuscated circuit on every possible key combination, and finally, an automated script is designed to insert the key gates. The framework is tested using circuits from benchmarks that include the ISCAS-85 suite [37]. For the fault-based encryption scheme, the is used to analyze the circuits for fault impacts along with the ABC tool [35] to generate the bench and Verilog files. By using the fault impact

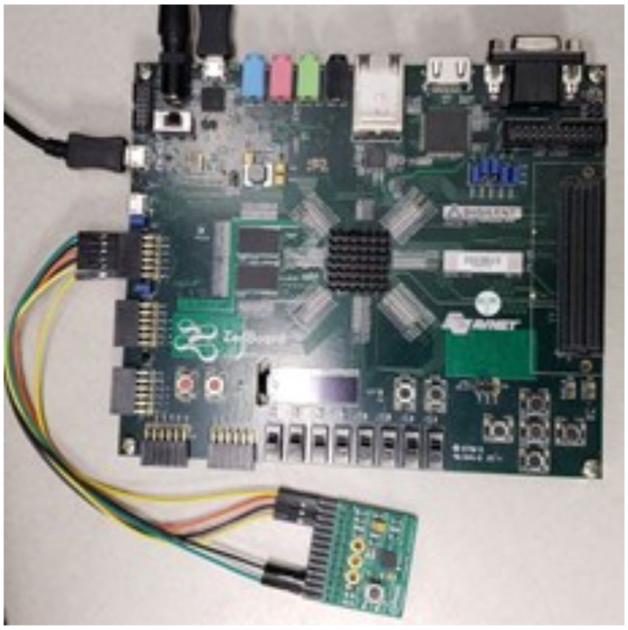


Fig. 6 Hardware setup

provided [38] in which using the stuck-at fault analysis the Fault impact = {(#test patterns detecting sa0) x (#output bits affected by sa0) x (#test patterns detecting sa1) x (#output bits affected by sa1)} is calculated. Key gates are inserted for the highest calculated fault impact which protects the IP from reverse engineering. This design offers flexibility to the design to control the corrupt outputs and to maximize design complexity for the attacker by target 50% Hamming distance with a smaller number of key gates which significantly reduces the area overhead.

Table 1 shows the fault impact summary for the ISCAS-85 benchmarks with different sets of test patterns. This shows the fault coverage along with the faults detected at different gates to compute the fault impact for logic encryption. Based on the fault impact factor Table 2 shows the average Hamming distance calculated for the benchmarks with a range of key sizes between correct and incorrect outputs to obtain a 50% Hamming distance with less number of keys to preserve the complexity of the locked design.

The whole framework is automated along with functional verification for the generated netlist using input test vectors. The key generated during logic obfuscation is stored on the server and upon mutual authentication, it is sent to the secure IP and stored on the TPM. The secure IP and AXI interconnect implemented by using the TEE is used to eliminate non-secure transactions and other AXI attacks. By using isolation design flow, the isolated secure IP block will have separate resources and ports for transactions. The master-slave ports with dedicated memory space are used for memory transactions (GP AXI Ports). The keys are stored in TPM with driver functions [39] and through authorized configuration, the application is unlocked.

Thus, unauthorized transactions and readback modifications are blocked by using the isolation technique. To secure the key from non-secure IP and AXI attacks, 128-bit AES encryption is done to the logic-locked key in the Secure IP. If any non-secure IP tries to access the key, only the encrypted version of the key will be available which makes it more secure. Figure 8 shows the serial terminal in which the secure IP gets the key from the server for the logic-locked application and saves it in the TPM and does 128-bit AES encryption to the key to camouflage the original key. This model provides security policies such as authorization by the user to update the system once the verification is done. Transactions with the Secure IP is not possible by any Non-Secure IPs(master) which eliminates illegal memory access. The AXI wrapper with custom IP creates a bridge between the PS and the PL. Configuration registers for the AXI ports are defined for the security policies of the application.

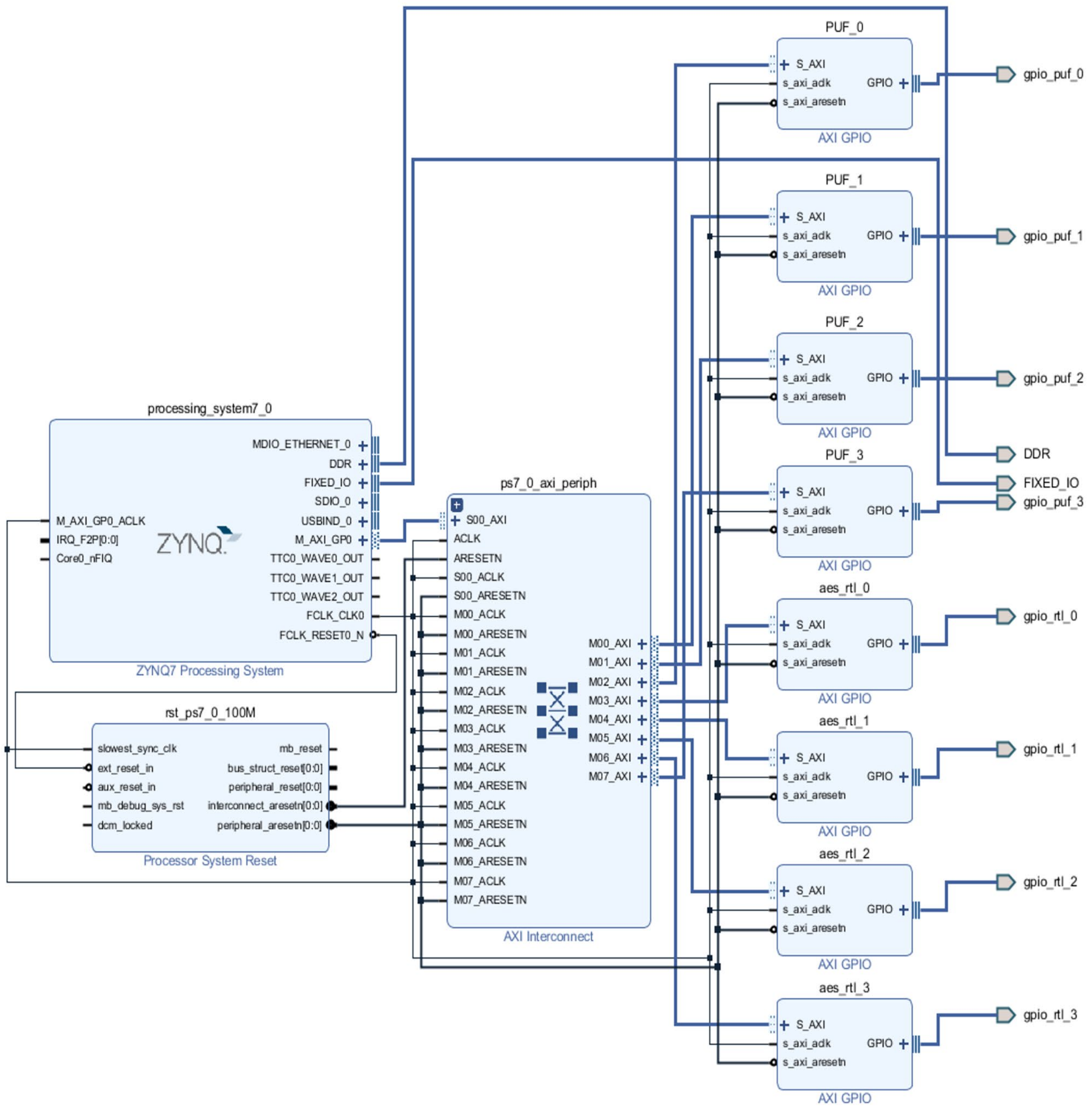


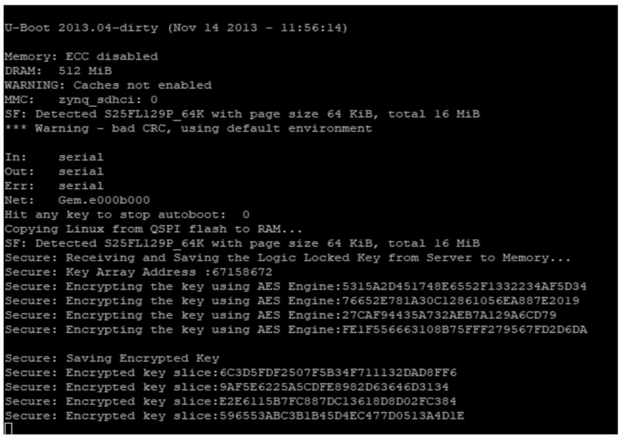
Fig. 7 System block with secure IPs

Table 1 Fault Impact summary for ISCAS-85 benchmarks

	C17	C432	C499
No of primary inputs	5	36	41
No of primary outputs	2	7	32
No of test patterns applied faults	224	224	224
No of detected faults	22	472	1271
No of undetected faults	0	28	83
Fault coverage	100 %	94.400%	93.9%

Table 2 Average Hamming Distance (50%) for the benchmark circuits with different key sizes

Range of Keys Size	Benchmark	Hamming Distance (%)
2-5	C17	50
17-20	C432	50
39-42	C499	50



```
U-Boot 2013.04-dirty (Nov 14 2013 - 11:56:14)

Memory: ECC disabled
DRAM:  512 MiB
WARNING: Caches not enabled
MMC:   zynq_sdhci: 0
SF: Detected S25FL129P_64K with page size 64 KiB, total 16 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
Copying Linux from QSPI flash to RAM...
SF: Detected S25FL129P_64K with page size 64 KiB, total 16 MiB
Secure: Receiving and Saving the Logic Locked Key from Server to Memory...
Secure: Key Array Address :67158672
Secure: Encrypting the key using AES Engine:5315A2D451748E6552F1332234AF5D34
Secure: Encrypting the key using AES Engine:76652E781A30C12861056EA887E2019
Secure: Encrypting the key using AES Engine:27CAF94435A732AEB7A129A6CD79
Secure: Encrypting the key using AES Engine:FE1F556663108B75FFF279567FD2D6DA

Secure: Saving Encrypted Key
Secure: Encrypted key slice:6C3D5FDF2507F5B34F711132DAD8FF6
Secure: Encrypted key slice:9AF5E6225A5CDFE8982D63646D3134
Secure: Encrypted key slice:E2E6115B7FC887DC13618D8D02FC384
Secure: Encrypted key slice:596553ABC3B1B45D4EC477D0513A4D1E
```

Fig. 8 Design using AXI GPIO Ports for Secure IP

6 Security Analysis

Secure boot is a root of trust process which holds all the keys used for cryptographic functions but if compromised leads to different malware attacks. Thus, implementing a secure boot with different security features for reconfigurable logic forms a resilient model against different attacks. The security properties of the proposed framework are:

- PUF-based challenge–response pairs for mutual authentication provides a unique set of key pairs or CRPS for authentication between the server and the device.
- PUF keys are used for decrypting the application bitstream, that is logic-locked. The logic-locking key is shared by the server after the authentication step is completed and are stored on a tamper-resistant memory of the device. The TPM module mitigates the invasive attacks to acquire the key.
- Logic-locked bitstream produces a corrupted output if the authentication fails which makes it unfeasible to clone the IP. The application's original functionality is unknown and difficult to break during runtime.
- The fault impact metric and Hamming distance analysis for fault-based encryption shows that with a smaller number of key gates the ambiguity of the locked design functionality is preserved, where any single incorrect key changes the functionality or the 50% of the outputs.
- IP isolation by TrustZone and wrapper implementation for secure IPs integrating AXI signals eliminates unauthorized transactions and readback modifications from FPGAs ICAP interface. AES encryption provides additional security by encrypting the keys and camouflages the original key hence protecting the keys from other access mediums.

7 Conclusion

In this paper, we present a secure framework to implement logic-locking and extend its application of secure boot process for FPGAs. The automated framework demonstrates the secure design flow to enable security functions such as RTL to secure bitstream, logic obfuscation, technology mapping, IP isolation, and support secure boot applications during runtime. The framework is tested using ISCAS 85, benchmark suite and is demonstrated on the Zynq 7000 family of Xilinx FPGAs. This framework is used for secure boot applications with authentication over the fly and secure IP transactions using TrustZone features.

References

1. Zhang Jiliang, Gang Qu (2019) Recent attacks and defenses on FPGA-based systems. ACM Trans Reconfigurable Technol Syst 12:1–24. https://doi.org/10.1145/3340557
2. Rajendran J, Sam M, Sinanoglu O, Karri R (2013) Security analysis of integrated circuit camouflaging. In: ACM/SIGSAC Conference on Computer and Communications Security, pp 709–720
3. Jarvis RW, McIntyre MG (2007) Split manufacturing method for advanced semiconductor circuits. US Patent 7(195):931
4. Chakraborty RS, Bhunia S (2009) HARPOON: an obfuscation-based SoC design methodology for hardware protection. IEEE Trans Comput Aided Des Integr Circuits Syst 28(10):1493–1502
5. Xilinx Inc. (2014) Zynq-7000 all programmable SoC secure boot. https://www.xilinx.com/support/documentation/user-guides/ug1025-zynq-secure-boot-gsg.pdf
6. Trusted Computing Group (2017) TCG PC client platform TPM profile (PTP) specification family '2.0' TCG public review
7. Benhani EM, Bossuet L, Aubert A (2019) The security of ARM TrustZone in a FPGA-based SoC. IEEE Trans Comput 68(8):1238–1248. https://doi.org/10.1109/TC.2019.2900235
8. Xilinx (2017) Understanding FPGA architecture. https://www.xilinx.com/html-docs/xilinx2017-2/sdaccel-doc/topics/devices/con-fpga-architecture.html/
9. Cowan C, Pu C, Maier D, Hintony H, Walpole J, Bakke P, Beattie S, Grier A, Wagle P, Zhang Q (1998) StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In: Usenix, p 5
10. Prandini M, Ramilli M (2012) Return-oriented programming, pp 84–87
11. Xilinx (2013) Zynq-7000 All Programmable SoC Software Developers Guide. Tech Rep. https://www.xilinx.com/support/documentation/user-guides/ug821-zynq-7000swdev.pdf
12. Xilinx and Inc. (2017) Using encryption and authentication to secure an ultraScale/ultraScale+ FPGA bitstream application note (XAPP1267), XAPP1267
13. Siddiqui AS, Gui Y, Saqib F (2020) Secure boot for reconfigurable architectures. Cryptography 4(4):26. https://doi.org/10.3390/cryptography4040026
14. Siddiqui AS et al (2019) Multilayer camouflaged secure boot for SoCs. In: 2019 20th international workshop on microprocessor/SoC test, security and verification (MTV), pp 56–61
15. Pocklassery G, Che W, Saqib F, Areno M, Plusquellic J (2018) Self-authenticating secure boot for FPGAs. In: 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, pp 221–226

16. Haj-Yahya J, Wong MM, Pudi V, Bhasin S, Chattopadhyay A (2019) Lightweight secure-boot architecture for RISC-V system-on-chip. In: 20th International Symposium on Quality Electronic Design (ISQED), pp 216-223. https://doi.org/10.1109/ISQED.2019.8697657
17. Hosseinzadeh S, Sequeiros B, Inácio PR, Leppänen V (2020) Recent trends in applying TPM to cloud computing. Secur Priv 3: n. pag
18. Zuo X, Liu W (2007) TPM based key backup and recovery. Int Conf Mach Learn Cybern 2007:2164–2167. https://doi.org/10.1109/ICMLC.2007.4370503
19. Shin J, Kim Y, Park W, Park C (2012) DFCloud: A TPM-based secure data access control method of cloud storage in mobile devices. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, pp 551–556. https://doi.org/10.1109/CloudCom.2012.6427606
20. Yasin M, Mazumdar B, Rajendran JJ, Sinanoglu O (2016) SARLock: Sat attack resistant logic locking. In: 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp 236–241
21. Yasin M, Mazumdar B, Rajendran JJ, Sinanoglu O (2017) TTLock: Tenacious and traceless logic locking. In: 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), p 166
22. Xie Y, Srivastava A (2018) Anti-sat: Mitigating sat attack on logic locking. IEEE Trans Comput Aided Des Integr Circuits Syst 38(2):199–207
23. Yasin M, Sengupta A, Nabeel MT, Ashraf M, Rajendran J, Sinanoglu O (2017) Provably-secure logic locking: From theory to practice. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp 1601–1618
24. Rajendran J, Pino Y, Sinanoglu O, Karri R (2012) Logic encryption: A fault analysis perspective. In: 2012 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp 953–958
25. Ngabonziza B, Martin D, Bailey A, Cho H, Martin S (2016) TrustZone explained: architectural features and use cases. In: 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC). Pittsburgh, PA, pp 445–451. https://doi.org/10.1109/CIC.2016.065
26. Pinto S, Santos N (2019) Demystifying arm trustZone: a comprehensive survey. ACM Comput Surv 51(6):1–36. https://doi.org/10.1145/3291047
27. Balisane RA, Martin A (2016) Trusted execution environment-based authentication gauge (TEEBAG). In: Proceedings of the 2016 New Security Paradigms Workshop (NSPW '16). Association for Computing Machinery. New York, NY, USA, pp 61–67. https://doi.org/10.1145/3011883.3011892
28. Zhao B, Xiao Y, Huang Y, Cui X (2017) A private user data protection mechanism in trustzone architecture based on identity authentication. Tsinghua Sci Technol 22(2):218–225. https://doi.org/10.23919/TST.2017.7889643
29. Gross M et al (2019) Breaking trustzone memory isolation through malicious hardware on a modern FPGA-SoC. Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop
30. Zhang J, Lin Y, Qu G (2015) Reconfigurable binding against FPGA replay attacks. ACM Trans Des Autom Electron Syst 20(2):1–20
31. Xilinx Inc., UG 470 - 7 Series FPGAs Configuration. https://www.xilinx.com/support/documentation/user-guides/ug470-7Series-Config.pdf. Accessed 20 Aug 2018
32. Chakraborty RS, Saha I, Palchaudhuri A, Naik GK (2013) Hardware trojan insertion by direct modification of FPGA configuration bitstream. IEEE Des Test 30(2):45–54
33. Aarestad J, Ortiz P, Acharyya D, Plusquellic J (2013) HELP: a hardware-embedded delay PUF. IEEE Des Test 30(2):17–25. https://doi.org/10.1109/MDT.2013.2247459
34. Zhang T, Wang J, Guo S, Chen Z (2019) A comprehensive FPGA reverse engineering tool-chain: from bitstream to RTL code. IEEE Access 7:38379–38389. https://doi.org/10.1109/ACCESS.2019.2901949
35. Berkeley Logic Synthesis and Verification Group (2005) ABC: a system for sequential synthesis and verification
36. Hyung Ki Lee and Dong Sam Ha (1996) HOPE: an efficient parallel fault simulator for synchronous sequential circuits. IEEE Trans Comput Aided Des Integr Circuits Syst 15(9):1048–1058. https://doi.org/10.1109/43.536711
37. Hansen MC, Yalcin H, Hayes JP (1999) Unveiling the iscas-85 benchmarks: A case study in reverse engineering. IEEE Des Test Comput 16(3):72–80
38. Rajendran J et al (2015) Fault analysis-based logic encryption. IEEE Trans Comput 64(2):410–424. https://doi.org/10.1109/TC.2013.193
39. Siddiqui AS, Saqib F. HEADS-UNCC/TPM-Baremetal-Drivers: TPM Baremetal for FPGAs and other Embedded Systems. https://github.com/HEADS-UNCC/TPM-baremetal-drivers. Accessed 30 Sept 2020