iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** I **Month of publication:** January 2025

DOI: https://doi.org/10.22214/ijraset.2025.66519

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue I Jan 2025- Available at www.ijraset.com

Review of Obfuscation Techniques in FPGA (Field Programmable Gate Array) Security

Nia Gella Augoestien, Abdul Ro`uf, Bambang Nurcahyo Prastowo, Jazi Eko Istiyanto
Departement of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta-Indonesia

Abstract: ***Hardware security is crucial for FPGA-based systems in addition to the functional realization of the system according to the expected specifications and optimal performance. FPGA has a longer supply-chain model that involves many entities. This results in greater opportunities for security threats and trust issues. Hardware Obfuscation is one method that can be used as a mechanism for a security effort from the threats of IP Piracy, Overbuilding, Trojan Hardware, Reverse Engineering, and Bitstream Modification. Obfuscation technique variations for FPGA-based system security cannot be fully implemented properly according to the specifications and limitations of the system design. To prevent and anticipate security threat attacks on FPGA-based systems, a performance analysis is needed to implement appropriate and effective techniques in system security. This paper discusses obfuscation techniques that can be implemented for security on FPGA-based systems according to the design process flow.***

Keywords: Hardware Security, Obfuscation, FPGA, Design Flow, Hardware Trojan

I. INTRODUCTION

This paper discusses obfuscation techniques that can be implemented for security on FPGA-based systems according to the design process flow. The use of compute hardware platforms such as Field Programmable Gate Arrays (FPGA) has been widely adopted in various application areas such as the Internet of Things, Cloud computing, automotive, and military electronics[1]. FPGAs used as the basis of the system vary from simple FPGAs to high-end FPGAs that have sophisticated features[2]. Although the main focus in developing FPGA-based systems is how to realize the functionality of the system according to the expected specifications and optimal performance, hardware security is crucial. The most attractive feature of FPGAs is the ability to reconfigure after the fabrication process to facilitate dynamic war needs. When compared to ASICs, FPGAs have advantages in terms of development time required, reliability, cost, and long-term maintenance[3].

Unlike Application Specific Integrated Circuit (ASIC), FPGA-based systems have a longer supply chain model that involves many entities [4] so the opportunity for security threats and trust issues to arise is greater. Forms of security threats to FPGA-based systems include device counterfeiting, Hardware Trojans, Reverse Engineering, IP Piracy, bitstream modification, and side-channel attacks [5]. The growing market share of FPGAs motivates an increasing number of attackers to target FPGA systems.

There have been many studies that propose various techniques that can be used to improve the security of FPGA-based systems, including bitstream encryption, Device identifier detection, Watermarking, PUF, Partial reconfiguration, and Hardware Obfuscation [4]. The implementation of these security techniques varies throughout the FPGA supply chain, where most of these techniques have been accommodated in high-end FPGAs that have sophisticated features. For some systems that use low-end FPGAs, these features are not yet equipped. Thus, this paper will discuss the use of Obfuscation techniques at each level of the FPGA design flow.

II. RELATED WORKS

With the increasing number of entities involved in the development of the supply chain and the design flow of FPGA-based systems, the security threats faced by FPGA-based systems are varied. There are 6 entities involved in the development of the FPGA supply chain, namely: FPGA Vendor, Foundry, IP Vendor, EDA tools vendor, FPGA-based system developer, and FPGA-based system user[4]. Each entity has its duties and roles. FPGA Vendor is a company such as Xilinx and Intel that designs the FPGA architecture that builds FPGA chips, while Foundry is a semiconductor factory that fabricates FPGA chips for FPGA Vendors. In the traditional design flow, these two entities are integrated into the same company, but along with the adaptation of global supply change in the development of Integrated Circuits that can minimize costs in terms of production and maintenance, in general, FPGA vendors and foundries are separate entities. Another entity involved in the FPGA supply chain is the IP Vendor who is tasked with developing IP Cores for specific applications on FPGAs. Examples of IP Cores can be DSP processors, cryptographic processors, and others. The use of IP Cores from certain IP vendors can shorten the development time of integrated circuits.

The short development time can also affect the overall cost. In developing an FPGA-based system, EDA tools are needed to facilitate circuit design and integration into a system. Several FPGA companies such as Xilinx and Intel complement the FPGAs marketed with software tools such as Vivado and Quartus to facilitate the development of FPGA-based systems. EDA tool development is sometimes carried out by vendors separate from the FPGA vendor.

Furthermore, FPGA can be utilized by developers of FPGA-based systems or also by users of FPGA-based systems. With the many entities involved, FPGA-based systems become vulnerable to security issues. Security techniques that can be used are based on variations in security threats that endanger FPGA-based systems. From the explanation given, several challenges and opportunities for research directions on FPGA-based system security can be seen, including the vulnerability of tools from FPGA Vendors used in the FPGA-based system design flow. In addition, it should be considered that the security of the FPGA bitstream is mandatory.

Bitstream is a binary configuration file loaded on the FPGA and includes the encoding architecture requirements to realize user designs[6]. In developing an FPGA-based system, several stages need to be passed by the developer of an FPGA-based system. The 5 general stages are system specification, system modeling, synthesis, implementation, and generating bitstream. At the system specification stage, the functional specifications of the system are defined along with several additional features. After the specifications are complete, the next stage is system modeling. FPGA-based system modeling can be done using HLS (High-Level Synthesis) or HDL (Hardware Description Language). If the system is modeled using HLS, it must first be compiled into RTL. The synthesis results obtained will then be implemented. At the implementation stage, it is divided into 3, namely mapping, place and route. The results of this implementation will later be converted into bitstream.

For the FPGA to be configured according to the bitstream that has been generated, several phases need to be passed. Threats from the FPGA bitstream perspective can be classified into 5 phases, namely Generation, Rest, Loading, Running, and end of life[2]. From the taxonomy of the forms of attacks described, it is known that several types of security threats can attack from more than 1 phase of the bitstream cycle such as IP Piracy, Trojan Hardware, and Reverse Engineering. Hardware Obfuscation is a powerful technique that can be used to prevent and anticipate these forms of attacks [7].

Hardware Obfuscation is a method used to modify a circuit to cover up hardware functionality so that incorrect key usage causes the device to not work according to its function[8]. The implementation of hardware Obfuscation techniques on FPGAs varies widely. However, these techniques cannot be fully adopted by simple FPGA-based systems. For example, in the research of[9], it was proposed to use random CFGLUTs programmed at system runtime on FPGA to perform Obfuscation. Although the results obtained are optimal in terms of security and overhead, this technique cannot be applied to simple FPGA-based systems that do not have these features.

Implementation of Obfuscation techniques can also be implemented at different stages in each FPGA design flow. 2 Obfuscation techniques at the RTL level that can be used to anticipate Hardware Trojans was propose by [10]. Implemented the Obfuscation technique by modifying the netlist of the logic synthesis process is designed by [3]. Implementing the Obfuscation technique in the high-level synthesis phase was proposed by [11].

The variations of the proposed Obfuscation techniques usually focus on specific design factors [12]. According to [13], several specific parameters can be used to measure the performance of the Obfuscation technique, including SAT-Hardness, Output Corruptibility, percentage of recoverable key bits, and additional Overhead both in terms of resource usage and execution time. The above considerations can be used to conduct further analysis in providing recommendations for effective obfuscation techniques for simple FPGA-based systems.

III. THREATS AND ATTACKS ON FPGA

Hardware Trojan is an act with malicious intent to modify electronic circuits or their design to change hardware functionality, leak confidential information, or destroy the system in a special condition design. Hardware Trojans can be inserted from IP modeling to the manufacturing process[10]. In general, hardware trojans can be divided into 2 parts, namely:

- Trigger is the part that is tasked with activating the payload when certain conditions are met
- Payload is the part that is tasked with carrying out hardware functions, leaking information, and reducing circuit performance.

Triggers are usually implemented in the remaining modules while payloads are distributed across many modules. [14]. Physically, the trigger module is implemented independently as a Single module that is not used in the original circuit. The trigger module will send different outputs depending on the conditions met. This condition is known as the Rare Condition. By placing the trigger process in Rare Conditions, it will be difficult to detect the presence of hardware trojans inserted into a system. Based on the elements contained in the trigger section, a hardware trojan can be classified into combinational Trojans and Sequential Trojans.

In combinational Trojans, there is no flip-flop element in the trigger section so the payload will be activated if the variation of input conditions is met. Triggers that are carried out when a transition to a certain state occurs using state elements are known as sequential trojans. The form of sequential trojans can also be distinguished into being triggered when a certain count value is met which is usually implemented using a k-bit counter. The second form will be triggered when a sequence of n-bit input values is met.
Reverse Engineering is the process of understanding the functional architecture of an electronic system using special methods. The purpose of reverse engineering on FPGA is to rediscover the high-level description of the bitstream. In general, the process can be done in 2 stages, namely decompiling the FPGA bitstream to produce a netlist and producing a high-level description of the obtained netlist. Bitstream in the early stages can be obtained from the 6th phase of the bitstream life cycle, while reverse engineering techniques on ASIC can be adopted when performing reverse engineering in later stages. On the other hand, reverse engineering on FPGA is often used to detect the presence of hardware trojans and IP hijacking. In 2019, a toolchain was proposed that was used to perform reverse engineering on the FPGA bitstream to produce RTL code which can later be used to detect hardware trojans. The technique used is to construct an integrated database that involves information from the FPGA architecture and bitstream mapping information. The results obtained are used to build the BRT (Bitstream Reversal Tools) and NRT (Netlist Reversal tools) tools to then combine the results obtained to obtain the RTL code. The evaluation results show that the tools created can rediscover the same RTL code and 100% correct netlist from the given benchmark bitstream [15].
IP piracy is a form of threat that uses licensed IP illegally. There are several forms of attack models, including obtaining IP in the form of a netlist by stealing the owner's IP encryption key, reverse engineering to obtain a netlist on an encrypted IP, using physical extraction techniques for encryption keys for encrypted bitstreams, and others. After obtaining IP illegally, changes are usually made to the LUT encoding, input pins on the LUT, LUT routes and placements, and the addition of dummy circuits to obscure the occurrence of IP piracy. In detecting IP piracy, there are several obstacles, including some commercial FPGAs that obscure the bitstream format and slight changes to the flow and design process will change the binary footprint of the bitstream[6].

IV. HARDWARE OBFUSCATION ON FPGA

Hardware Obfuscation is a method used to modify the design, making it difficult to reverse engineer or piracy [7]. This form of protection is an active category that transforms the Original design into a disguised design through functional and structural transformations. Obfuscation results work with 2 modes, namely Lock and Unlock. Unlock mode is obtained when the use of IP uses the correct key so that the circuit works according to its function, while when the wrong key is used, the IP is in Lock mode and the circuit does not produce the correct function. Many Obfuscation techniques can be implemented on FPGA. In this paper, obfuscation techniques are explained based on the design stage flow. Behavioral modeling is the initial stage in the FPGA design flow. Obfuscation carried out at this stage is more flexible because the disguised device can be used for different needs [16]. The Obfuscation process begins with structural modeling of the circuit, then identifying the modules in the circuit that have the most impact on the output of the circuit and the next level circuit. The identified modules will be Obfuscated. Thus the Obfuscation process is not performed on every module in the circuit, but only on one module. This design method produces low overhead in terms of resource usage for the Obfuscation circuit.
The Obfuscation process at the RTL level in FPGA can be done by adding a state with the correct key sequence to a circuit. If the key sequence is wrong, then the final state will never be reached. However, if the key sequence is correct, then the module works according to its function because the obfuscated circuit will reach the state where the original circuit operates [10]. By using this scheme, the Obfuscation results can prevent the obfuscated module from using a black box with an overhead that is not too significant. However, the obfuscation mechanism which is designed structurally separate from the actual functional mode of the circuit makes the circuit identifiable using reverse engineering. The obfuscation approach by utilizing the bit content used in the LUT can be implemented at the Netlist level in FPGA, thereby reducing overhead in terms of Delay, Power, and Area. The initial stage required is to identify all LUTs that do not utilize all of their inputs. For example, if the FPGA architecture has a 6-input LUT, then all LUTs that only utilize 5 or fewer inputs can be used. Next, prioritize the LUT candidates that can be used based on the highest fan-out so that it produces a large corruptibility output[3]. Some things to consider from using this method are that it is not suitable for implementation in certain situations where there are not enough LUTs that have free space to use. In addition, in very complex circuits, there is a possibility that the implementation results cannot be routed for design on FPGAs with high constraints and very deep logic levels.
Obfuscation method using Configurable look Up Tables (CFGLUT). CFGLUT is entered randomly based on optional parameters in the design. Then program the CFG to complete the FPGA bitstream and design functionality. There are 2 stages, namely: Selecting the part of the HDL that will be configured using CFG based on considerations either randomly or Some parts that are vulnerable to

HDL code attacks. Furthermore, using CFGLUT to substitute the HDL part to complete the bitstream during the running process[9]. This method can only be implemented on FPGA architectures that have CFGLUT features, namely on high-end FPGAs that can be partially configured.

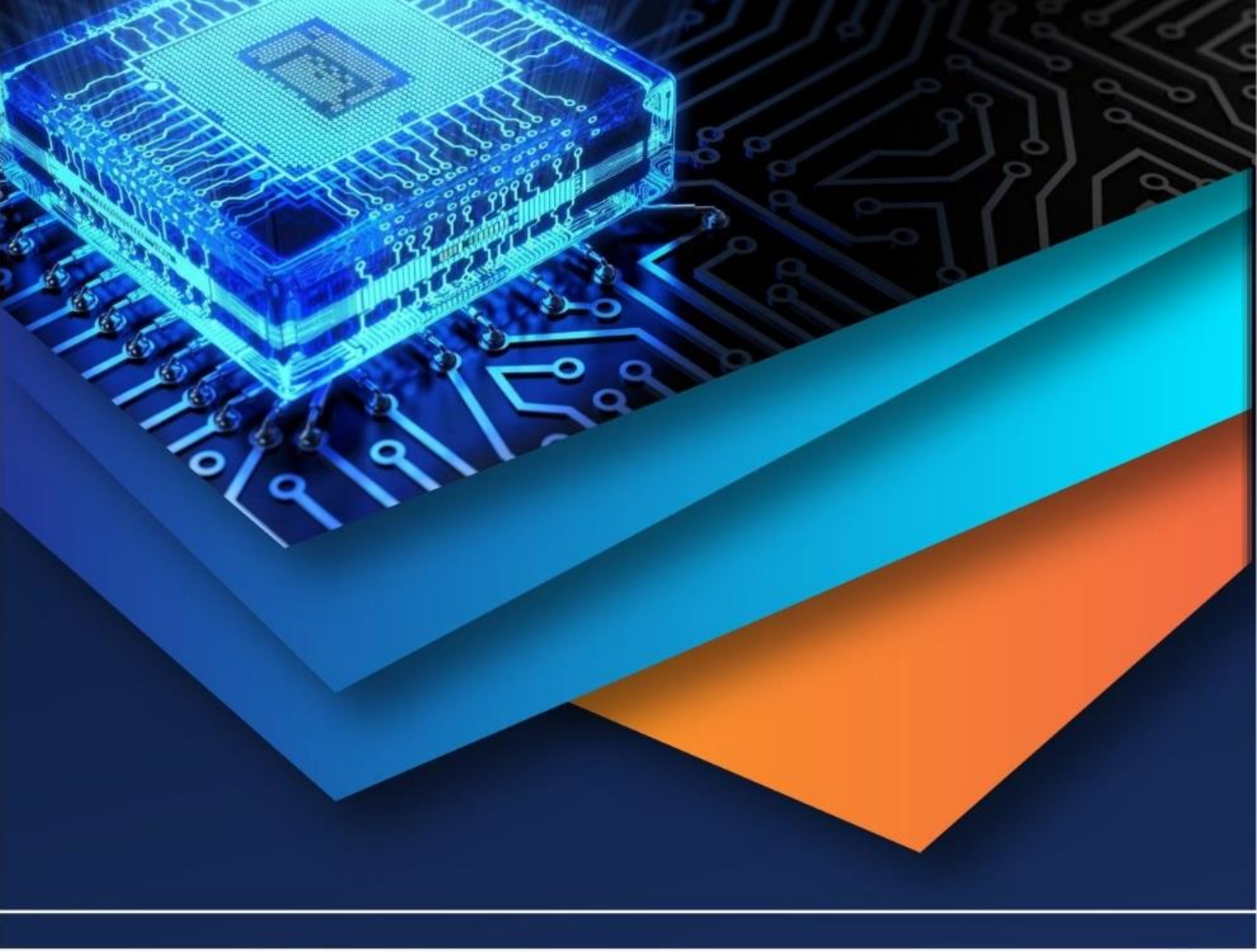
In its application, Hardware Obfuscation to secure systems on FPGAs often collaborates with PUF (Physical Uncloned Function). PUF is used to generate keys that will be used for obfuscation. Without the correct key generated by the PUF, the functional system embedded in the FPGA cannot work properly. The key generated from PUF is UNCLONE because it depends on the variation of the IC manufacturing process. so that with the same input it will produce different outputs on each different chip. The characteristics of the key generated by PUF are also unpredictable because they vary on each IC. So when using PUF to generate keys for obfuscation, 2 schemes are needed that can be done with key knowledge before designing obfuscation. Based on the key generated by PUF, the obfuscation technique is implemented. Designing an obfuscation technique whose key can be flexible. The key used does not change the flow of the obfuscation process that is carried out.

V. CONCLUSIONS

Obfuscation technique variations for FPGA-based system security cannot be fully implemented properly according to the specifications and limitations of the system design. To prevent and anticipate security threat attacks on FPGA-based systems, a performance analysis needs to be carried out to implement appropriate and effective techniques in system security. The higher the level of implementation of the Obfuscation technique, the more significant the results of the structural transformation produced, but the use of automatic EDA tools in the design flow can eliminate the obfuscation part in the optimization process. Collaboration of Obfuscation Techniques with other security methods can optimize circuit security.

REFERENCES

[1] B. Olney and R. Karam, "WATERMARCH: IP Protection Through Authenticated Obfuscation in FPGA Bitstreams," in IEEE Embedded Systems Letters, vol. 13, no. 3, pp. 81-84, Sept. 2021, doi: 10.1109/LES.2020.3015092.
[2] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi and M. Tehranipoor, "FPGA Bitstream Security: A Day in the Life," 2019 IEEE International Test Conference (ITC), 2019, pp. 1-10, doi: 10.1109/ITC44170.2019.9000145.
[3] B. Olney. and R. Karam. 2020. Tunable FPGA Bitstream Obfuscation with Boolean Satisfability Attack Countermeasure. ACM Trans. Des. Autom. Electron. Syst. 25, 2, Article 19 (January 2020), 22 pages. https://doi.org/10.1145/3373638
[4] J. Zhang and G. Qu. 2019. Recent Attacks and Defenses on FPGA-based Systems. ACM Trans. Reconfgurable Technol. Syst. 12, 3, Article 14 (August 2019), 24 pages. https://doi.org/10.1145/3340557
[5] S. Sunkavilli, Z. Zhang and Q. Yu, "New Security Threats on FPGAs: From FPGA Design Tools Perspective," 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2021, pp. 278-283, doi: 10.1109/ISVLSI51109.2021.00058
[6] G. Skipper, C. Sozio, A. Duncan, A. Lukefahr and M. Swany, "ReCon: From the Bitstream to Piracy Detection," 2020 IEEE Physical Assurance and Inspection of Electronics (PAINE), 2020, pp. 1-6, doi: 10.1109/PAINE49178.2020.9337563.
[7] S. Bhunia, M. Tehranipoor, 2019, “Hardware Security: A Hands-on Learning Approach”, Morgan Kaufmann, United States.
[8] S. Chhabra and K. Lata, "Design and Analysis of Logic Encryption Based 128-Bit AES Algorithm: A Case Study," 2018 15th IEEE India Council International Conference (INDICON), 2018, pp. 1-6, doi: 10.1109/INDICON45594.2018.8987098.
[9] M. Labafniya,, & S. E. Borujeni, 2021. An obfuscation method based on CFGLUTs for security of FPGAs. ISeCure, 13(2), 157-162. doi:10.22042/isecure.2021.234848.557
[10] N. Giri and N. N. Anandakumar, "Design and Analysis of Hardware Trojan Threats in Reconfigurable Hardware," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-5, doi: 10.1109/ic ETITE47903.2020.227.
[11] Islam, S. A., Sah, L. K., & Katkoori, S. (2021). High-level synthesis of key-obfuscated RTL iP with design lockout and camouflaging. ACM Transactions on Design Automation of Electronic Systems, 26(1) doi:10.1145/3410337
[12] Kolhe G, Sai M. PD, Rafatirad S, Mahmoodi H, Sasan A, and Homayoun H. 2019. On Custom LUT-based Obfuscation. In Proceedings of the 2019 Great Lakes Symposium on VLSI (GLSVLSI '19). Association for Computing Machinery, New York, NY, USA, 477–482. https://doi.org/10.1145/3299874.3319496
[13] M. Yasin, J. Rajendran, O. Sinanoglu, 2020, “Trustworthy Hardware Design: Combinational Logic Locking Techniques,“Springer : Switzerland.
[14] M Cho, J Jang, Y Seo, S Jeong, S Chung, T Kwon, Towards bidirectional LUT-level detection of hardware Trojans, Computers & Security, Volume 104, 2021
[15] T. Zhang, J. Wang, S. Guo and Z. Chen, "A Comprehensive FPGA Reverse Engineering Tool-Chain: From Bitstream to RTL Code," in IEEE Access, vol. 7, pp. 38379-38389, 2019
[16] K. Saravanan and N. Mohankumar, "Design of Logically Obfuscated n-bit ALU for Enhanced Security," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019, pp. 301-305, doi: 10.1109/ICECA.2019.8822129



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY