International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
OPEN ACCESS
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Implementation of Logic-Locking Technique Based on Probability Using Back End Tool

P. Rajesh[1], Sompalli Charan Sai[2], Vellala Sai Sri Pranathi[3], Udatha Kavya Sree[4], Thippareddy Asuvardhan Reddy[5], A R Kushal[6]

[1]Assistant Professor, Department of Electronics and Communication Engineering, SV College of Engineering (SVCE), Tirupati, A.P. India
[2, 3,4,5,6] UG Students, Department of Electronics and Communication Engineering, SV College of Engineering (SVCE), Tirupati, A.P. India

ARTICLEINFO

Article History:

Accepted: 10 March 2024
Published: 20 March 2024

Publication Issue :
Volume 11, Issue 2
March-April-2024
Page Number :
262-272

ABSTRACT

Integrated circuit (IC) piracy and overproduction are serious issues that threaten the security and integrity of a system. Logic locking is a type of hardware technique where additional key gates are inserted into the circuit. Here probability-based logic-locking technique to protect the design of a circuit. Our proposed technique, called "ProbLock", can be applied to both combinational and sequential circuits through a critical selection process. We have to use filtering process to select the best location of key gates based on various constraints. Each step in the filtering process generates a subset of nodes for each constraint. The Probability-Based Logic-Locking Technique is a security measure that aims to protect the confidentiality and integrity of integrated circuits. This technique uses a combination of DSCH and MICROWIND tools to generate logic-locked designs that are resistant to reverse engineering attacks. The logic-locking process involves adding additional gates to the design, which are controlled by secret keys, and thereby obfuscating the original circuit's functionality. The probability-based approach introduces randomness in the process, making it difficult for attackers to determine the correct key. By using a stochastic algorithm, the locking mechanism generates a set of gates that have a probability distribution based on the secret key. The resulting design is then verified for correctness and functionality using MICROWIND tools. This abstract presents a novel technique for generating logic-locked designs using DSCH and MICROWIND tools with a probability-based approach. The technique aims to provide increased security for integrated circuits, making them less vulnerable to reverse engineering attacks. The proposed technique is evaluated using simulations and experimental results, which demonstrate the effectiveness of the approach in preventing unauthorized access to sensitive information stored in the circuit.

Keywords : DSCH and MICROWIND, Probability based Locking

I. INTRODUCTION

The semiconductor industry is constantly changing, from the production of integrated circuits (IC)s to the complexity of their design. The industry has moved to a fabless model where most of the fabrication for a chip is outsourced to a less secure and less trusted environment. From intellectual property (IP) design to manufacturing, an IC has go to through an extensive process before it reaches the end user [1]. The supply chain stages are shown in Figure 1. First, the IP owner designs a module at the RTL level, gate level, and layout level. Multiple IP designs get integrated onto a single system on chip (SoC). Next, a foundry fabricates the IC die and an assembly will package the die with pins and wires into a complete package. The final package gets manufactured and distributed out to end users and consumers. These environments in the supply chain include testing and fabrication facilities that are necessary for the pipeline. Testing and fabrication facilities are usually outsourced to other countries where it is cheaper to finish the work. While this model does improve production costs and development, it has also led to the consequence of piracy, overproduction, and cloning. An IP owner does not have control over these un trusted facilities so IP piracy is a common issue. The chips are also vulnerable to various attacks that attempt to extract the design of the chip or other information from the device. Due to these security issues, researchers have developed techniques to counter these attacks. Other research topics including developing attacks to evaluate the security and privacy of ICs at different stages of the supply chain.

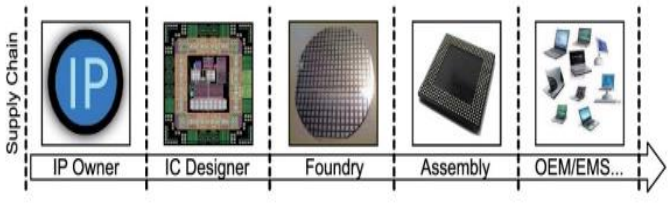


Figure 1: IC Supply Chain Stages [1]

The main threats in the IC supply chain are reverse engineering, IP piracy, and tampering [1]. The main goal of reverse engineering is to determine the design and behavior of the IP modules on an IC or SoC. Reverse engineering attacks will exploit the weaknesses and security vulnerabilities of an IC to recreate the original design of an IP. The reverse engineered design can then be used to sell counterfeit hardware on the black market. Reverse engineering can occur anywhere along the IC supply chain including the design house, foundry, testing site, and at the end user. Reverse engineering attacks are also usually destructive and sometimes require chemical and physical alteration of the IC to recover the design. IP piracy is another major concern in the IC supply chain. Facilities in the IC supply chain that use the IP illegally are violating piracy rules. Piracy usually includes overproduction and counterfeit production. A foundry with access to the IP from the designer can produce more ICs than what was ordered and sell the extra for profit. Adversaries at these sites can also clone or copy the design and sell that for profit as well. Tampering is the idea of modifying the design for other than its intended purpose. An attacker can accomplish this by inserting hardware Trojans that will exploit sensitive data on the IC and transmit that data to an outside party [2]. Trojans can also sabotage the IC by targeting critical path data such as power and timing modules on the circuit.

This paper is organized in five sections. After this introduction, in Section II, literature survey discussed of the paper, section III about the Existing system, Section IV about Proposed System, as well as the novel feature of the proposed method. Finally, Sections V and VI provide the simulation results and the conclusions and Future work, respectively.

.

II. RELATED WORK

Many techniques of logic locking have already been proposed and tested against certain attacks and on circuit benchmarks. One of the earliest logic-locking

techniques inserted key gates randomly into the circuit. This provided some security, but many attacks were developed to break this method. Another obfuscation technique was developed using logic-cone analysis in [3].

Sections of a circuit can be grouped into logic cones by calculating the fan-in and fan-out values of a gate. Inserting key gates at certain logic cone areas will increase the security of the system. Logic-cone analysis is good for countering logic-cone attacks. Certain attacks will exploit these weak logic cones and try to discover the key to unlock the circuit. Logic-cone analysis is vulnerable to other types of attacks such as SAT and functional attacks. Strong logic locking (SLL) is another obfuscation technique, but it is also vulnerable to SAT attacks [4].

SLL is based on interference graphs that show how inserted key gates interfere with each other. The interference graph shows the relationship between an inserted key gate and its surrounding key gates and wires. The interference graph shows if key gates are on a cascading path or parallel path, or if they do not interfere with each other at all. The interference graph along with other information makes it harder for an attacker to unlock the circuit even with SAT attack models. SLL was initially evaluated with a hill-climbing attack where the bits of an initial random key guess is toggled to minimize hamming distance between circuit outputs and test responses. If a key produces a hamming distance of 0, the attack is considered successful. SLL was compared against random logic locking and a fault-based technique. It was shown that the hill-climbing attack was ineffective at determining the correct key value for all tested ISCAS ’85 benchmarks while also being able to break some of the random locked circuits.

More recent techniques have been developed to counter SAT attacks and other related schemes. The obfuscation technique needs to be strong enough to resist certain attacks; otherwise the integrity of the IC would be compromised. The goal of an adversary during an attack is to determine the secret key to unlock the circuit or gain other important information from the system. SARLock was developed to make the SAT attack model inefficient [6].

SARLock employs a small overhead strategy that exponentially increases the number of distinguishing input patterns (DIPs) needed to unlock the circuit. SARLock is very strong against SAT attacks since it uses the basis of the attack model to determine where to insert key gates. The input pattern and corresponding key values can be analyzed during the insertion process of the obfuscation technique. SarLock was evaluated using a SAT attack and calculating the number of DIPs needed to determine the correct key value to unlock a locked circuit.

A subset of the ISCAS ’85 benchmarks were encrypted with SarLock and SLL and then evaluated with a SAT attack. SarLock proved that it was more effective against SAT attacks because it took a larger number of DIPs and more time to break the circuit. The SAT algorithm would run for hours to break a SarLock circuit, but it took less than a second for all SLL circuits. In 2017, TTLock was proposed, which resisted all known attacks including SAT and sensitization attacks [10].

TTLock would invert the response to a logic cone to protect the input pattern. The logic cone would be restored only if the correct key is provided. The small change to the functionality of the circuit would maximize the efforts needed for the SAT attacks. The generalized form of stripping away the functionality of logic cones and hiding it from attackers is known as stripped-functionality logic locking (SFLL). However, the design of the TTLock did not account for the cost of tamper-proof memory, which could lead to high overhead in the re-synthesis process [11,12].

Another group automated the general process of TTLock to identify the parts of the design that needed to be modified in an efficient way. They used ATPG tools to develop a scalable and more efficient way of protecting these patterns from attackers.

Overall, a 35% improvement in overhead was achieved with the automated process. Later, a modified version of SFLL was proposed based on the hamming distance of the key. This was referred to as SFLLhd [13].

The hamming distance metric was used to determine which pattern to modify in the SFLL scheme. Depending on the type of attack, the hamming distance can be adjusted accordingly. In 2019, the idea of exploring high-level synthesis (HLS) with logic locking was proposed with SFLL-HLS [14].

SFLL-HLS was proposed to improve the system-wide security of an IC. The design resulted in faster validation of design and higher levels of abstraction. The HLS implementation in this technique was used to identify the functional units and logic cones to be operated on with respect to SFLL. They observed low overhead and power results from their analysis. The strength of SFLL was evaluated in [13] where a SAT-based attack was developed against SFLL-HLS and other SFLL techniques. Similar to most logic locking techniques, SFLL is vulnerable to strong SAT attacks. The group used synthesized RTL circuits, which were smaller than public benchmark suits from ISCAS ’85 and ISCAS ’89. The SAT attack was able to determine the correct key within seconds for all of these benchmarks. Most recently in 2020, LoPher was developed as another SAT-resistant obfuscation technique [15].

LoPher uses a block cipher to produce the same behavior as a logic gate. The basic component for the block cipher is configurable and allows many logic permutations to occur, which further increases the security of the system. In 2020, another group presented a scalable attack-resistant obfuscation logic locking technique (SARO) [16].

SARO splits circuit benchmarks into smaller partitions and performs a systematic truth table transformation (T3) for each partition. The T3 process is highly randomized and leads to highly altered structural and graphical representation of the circuit netlist. The first step of SARO is to partition the circuit into a smaller area and analyze the fan in the cone as well as the logic depth of a partition. The fans in cones that cover more inputs are more suitable for the functional transformation of the T3 process. Similarly, a high logic depth allows for more random transformation in structural and graphical representation. Depending on the type of logic gate, selected inputs, and key bit inputs, the T3 process transforms the boolean logic of a gate to add functional obfuscation to the design. The proper key value allows gate logic to function as normally designed. SARO was evaluated against a SAT attack, and for all encrypted ISCAS ’85 benchmarks, the SAT attack was unable to determine the correct key value. SARO is extremely effective against power attack schemes such as a SAT attack; however, it suffers from a high complexity and large overhead. The T3 process has an exponential amount of functions to evaluate, and complexity can vary with number of inputs and key size. The area overhead of SARO is 23%, which is twice as large as most presented obfuscation techniques. Many forms and variations of SAT attacks have been created in order to show the weaknesses of various hardware obfuscation techniques. Algorithms have been developed for SAT competitions, and the results can be used in a variety of applications including hardware obfuscation [17].

These tools are used to evaluate the strength of logic locking techniques and can be used to bypass the security of integrated circuits. As a result, an anti-SAT unit was developed as a general solution to the SAT attack [18].

III. PROBABILITY BASED LOGIC LOCKING

Probability based logic locking or ProbLock is a novel functional logic locking technique based on filtering out nodes in a circuit to find the best location to insert key gates. The final stage of the filtering process uses the probability of gate outputs to determine where to insert key gates. ProbLock is an algorithm where the key gates are either XOR or XNOR gates and a key is

used to unlock the circuit. We used four constraints to determine the best candidate nodes to insert our XOR or XNOR key gate; longest path, non-critical path, low dependent nodes, and best probability nodes. The first three constraints find the set of nodes that lie on the longest path, non-critical path and have low dependent wires. The last constraint uses probability to find the set of nodes equal to the key length where we will insert the key gates. We chose the longest path and non-critical path constraint to avoid critical timing elements and to insert key gates on parts of the circuit that was being used the most. We chose the low dependent wires and probability constraint to determine locations where the output would be changed the most. This would make it harder for an attack to generate the golden circuit using an oracle based attack. Once we determine the location of the key nodes, we can insert key gates into the netlist and re-synthesize the circuit. In Equation 1 the candidate nodes are determined from a function of all four constraints. LP is the set of nodes on the longest paths while NCP is the set of nodes on non-critical paths. LD represents the set of low dependent nodes and P are the set of probability nodes

$$\text{Selected Nodes} \subset P \subset LD \subset NCP \subset LP$$

For our obfuscation technique, we decided to lock a set of combinational and sequential circuit netlists using the ISCAS ’85 and ISCAS ’89 circuit benchmarks [19] [20]. We obfuscated a total of 40 benchmarks using ProbLock. For some of the constraints, we had to use an unrolling technique described in [21] to accurately filter out nodes. This unrolling technique was only used in sequential circuits to simplify the concepts of flip flops and other sequential logic. The sequential logic can be replaced by the main stage and a k number of sub-stages depending on the number of times unrolled. This results in a k-unrolled circuit that has the same functionality as the regular circuit. For this process, we generated a set of unrolled ISCAS ’89 benchmarks which we used in some constraint algorithms. We unrolled these circuits once to prevent inaccuracies in constraints such as the longest path and non-critical path. Finally, we integrated a custom low overhead anti-SAT block based on an established anti-SAT block method [24].

A. LONGEST PATH CONSTRAINT

The longest path constraint isolates a subset of nodes that lie on the longest paths in a circuit netlist. The subset of nodes is different for each circuit and is a function of the key length determined for each circuit. We represent the netlist of each benchmark as a directed acyclic graph (DAG) and perform the longest path analysis on each DAG. Each vertex in the DAG is a gate element from the netlist and each vector represents the wire connecting to the next gate element. Once the DAG is constructed for each benchmark, we calculated the longest paths of the DAG using a depth first search (DFS) technique. We then calculate the next longest path to generate a subset of nodes along the longest paths. Each unique node in the longest path gets added to a subset during each iteration until the size of the subset is bigger than two times the key length for that circuit. The structure of this theory is shown in Algorithm 1 which uses the DFS in Algorithm 2. Figure 12 shows the longest path for the circuit to be 3 since there are 3 gates between input A and output Y.

The next longest path would also be 3 from input B to output Y. All of the nodes along both longest paths would be added to a subset of the longest path nodes. Once this subset of longest path nodes is determined, that subset gets used in the next filtering constraint. This subset can be adjusted to include more or fewer nodes depending on other filtering constraints. If more nodes are needed, this constraint is the first to be modified. We chose to use the longest path constraint to counter oracle guided attacks. Oracle guided attacks will query the IC with various inputs and observe the output. This gives the attacker

information about how the circuit behaves and the adversary can use this information to determine the secret key. We want to insert key gates where most of the logic and activity occur in the circuit. An oracle guided attack will most likely pass data through the longest paths of a circuit so we want to protect these parts of the IC by inserting key gates on the longest path

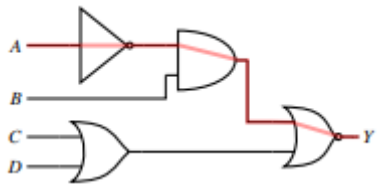


Fig.1: Longest Path (in red)

B. CRITICAL PATH CONSTRAINT

The critical path constraint is similar to the longest path; however, rather than considering logic depth, we look at timing information. This constraint is essential, as adding gates on the critical path could break the circuit functionality or change timing specifications. We used Synopsys Design Compiler (DC) [22] to compile the benchmarks and calculate the critical paths. In DC, the critical path of a circuit can be determined with several simple steps. First, the benchmark in Verilog format is imported into the software. Next, the delay timing and timing constraints are set for certain specifications. Finally, DC will calculate the critical path from a primary input to a primary output. The results are a set of nodes that represent the critical path. For this step, ProbLock calculates multiple critical paths and exports the data to be used in the filtering process. Figure 13 shows an example of the timing analysis of a basic circuit. The figure shows that the two critical paths lie from input C to output Y and from input D to output Y. The propagation timing for the critical path is 60ps as opposed to 65ps on the other paths. The nodes selected often overlapped with other constraints (e.g. the longest path was often the critical path), though oftentimes the critical path would involve gates with large fan out. Determining the critical path is largely technology-specific; different process design kits (PDKs) will have different timing information which can affect which paths are critical paths. We removed any nodes that were on the critical path from the set of nodes passed into this constraint. The resulting subset results in nodes that are on the longest, non-critical path.

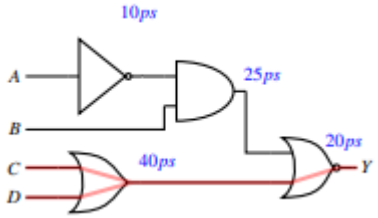


Fig.2: Critical Paths (in red)

C.LOW WIRE DEPENDENCY

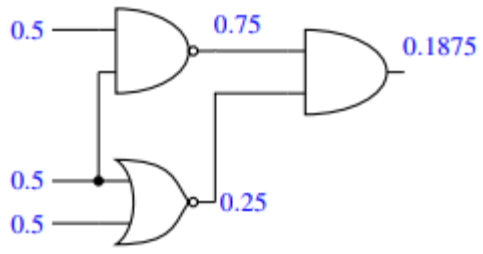
Constraint The next constraint generates a subset of nodes that are connected to low dependent wires. The output wire of a gate is considered low dependent if the input wires to that gate have little influence on the value of output. This idea is modified from a technique called FANCI where suspicious wires can be detected in a Trojan infected design [25]. A functional truth table is created for each output wire of each gate in the circuit. The inputs of the truth table correspond to the inputs of the gate being analyzed. For each input column, the other columns are fixed and each row is tested with a 0 or 1 to determine the output. This results in two functions when setting the value to either 0 or 1. The boolean difference between these two functions results in a value between zero and one that can be further analyzed. The value for each input gets stored as a list for each output wire. We take the average value of the entire list to determine the dependency of an output wire.. This analysis can determine if certain inputs are low dependent or if they rarely affect the corresponding logic. Low dependent wires are weak

spots in the circuit so this constraint isolates those locations to improve the security

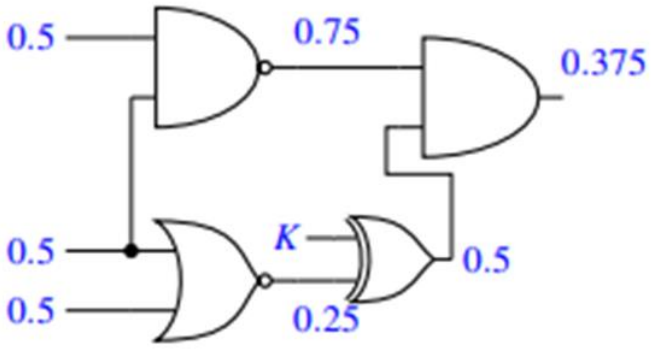
D. BIASED PROBABILITIES CONSTRAINT

The probability constraint focuses on reducing the effectiveness of the SAT attacks. In a SAT attack, a distinguishing input (DI) is chosen and the attacker runs through various key values, eliminating any which yields an incorrect output. Thus, to reduce the effectiveness of a SAT attack, the number of wrong keys produced for a given DI must decrease. This can be done by bringing the probability of any given node being 1 closer to 0.5, since any node which is biased towards 0 or 1 will propagate through to the output nodes, making it easier for SAT attacks to eliminate key values. Since a two-input XOR/XNOR has an output probability of 0.5, we can insert our key gates at nodes heavily biased towards 0 or 1 and "reset" the probability to 0.5. The algorithm used to obtain the N nodes with the most biased probabilities is shown in Algorithm 4. The output probabilities of a gate are dependent on the type of logic gate and the probabilities of its inputs.

The probability of incoming input wires will influence how the output probability is calculated for each gate type. It is worth noting that while generating node probabilities for combinational circuits is trivial, sequential circuits pose a potential problem because of the D flip flops (DFFs). However, giving the DFF outputs a starting probability of 0.5 and propagating running a few iterations (three is sufficient) will asymptotically approach the correct probability for the DFF node.



(a) Pre-Insertion Probabilities



(b) Post-Insertion Probabilities

Fig.3: Key Gate Insertion Probabilities

An example of this is illustrated in Figure 3. Figure 3 shows a sample circuit with each node annotated with the probability of that node being logic 1. The output shown is heavily biased toward logic 0, which makes it more susceptible to SAT attacks. Strategically adding a key gate, as shown in Figure 16 b brings the output probability closer to 0.5, reducing the effectiveness of the SAT attack. This idea of using probability calculations for logic locking is a novel concept that has not been tested yet. After the first three filtering constraints, the ProbLock algorithm has a set of nodes that lie on the longest path, non critical path and are low dependent. From this set of nodes, we generate a new subset equal to the bit size of the key value based on the biased probabilities of each node. The final set of nodes should produce the best location to insert key gates for a circuit netlist. This is the final step for the ProbLock algorithm that results in a low overhead locked benchmark. E. Anti-SAT ProbLock is a low overhead logic locking technique that aims to counter SAT based attacks on integrated circuits. However, other anti-SAT techniques have been developed to be provably effective against newer SAT attacks. The lightweight anti-SAT block (ASB) in [26] is a modification of the regular anti-SAT block in [24] that reuses overlapped key gates from the logic locking technique in the new lightweight anti-SAT block. The regular ASB described in [24] is a low overhead module that can connect to wires and key inputs of the original locked

circuit. This block exponentially increases the attack time and iterations of a SAT attack. The ASB is composed of additional key gates and logic components that are integrated with the original logic of the locked circuit.

The lightweight ASB minimizes overhead by creating an ASB that reuses the key gates from the original locked circuit and only adds the additional key gates to complete the full ASB. This method reduces the overhead of the ASB while maintaining the same level of security against SAT attacks. In Figure 17a, an example circuit is shown. Figure 17b shows the locked version of that circuit by inserting key gates E1, E2, and E3. In Figure 17c, the lightweight ASB implementation is shown by reusing key gates E1, E2, E3, and adding gates E4, E5, E6, G4, and G, to create the full ASB. For this specific example, the overhead is reduced by almost 50% and maintains the effectiveness against a full SAT attack. We used the same idea to integrate a strong anti-SAT solution into ProbLock. During the ProbLock algorithm, the best nodes are selected to be candidates for inserted key gates. After the key gates are established, the algorithm parses for patterns that would be suitable to construct a lightweight ASB. If a combination of key gates can be used to create a lightweight ASB, those nodes are flagged until later. Once all of the overlapped nodes are determined, the final step of ProbLock constructs a final locked circuit with an integrated ASB.

IV. PROPOSED METHOD

A. *UNLOCKED CIRCUIT*

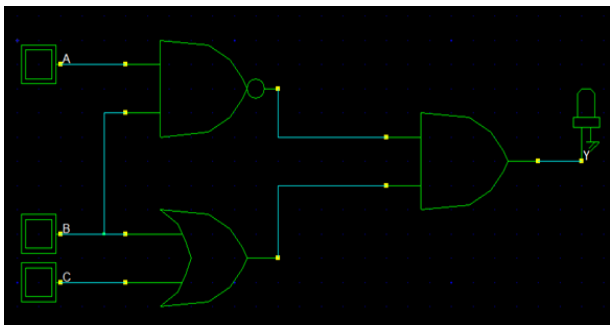


Fig.4 : Unclocked Circuit Design using DSCH Tool

The use of NAND, NOR, and AND gates in an unclocked circuit does not directly involve probability, as these gates are deterministic in nature. However, the behavior of an unclocked circuit can be analyzed probabilistically by considering the probabilities of different input combinations and their corresponding output probabilities.

B. LOCKING CIRCUIT

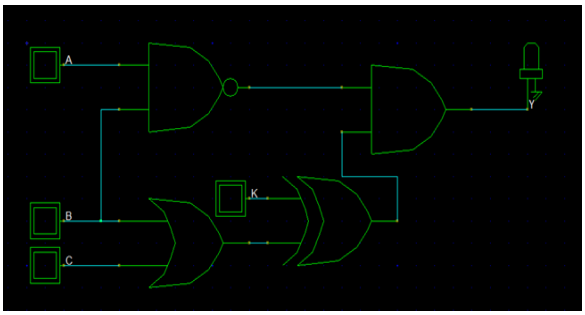


Fig.5 : Locking Circuit Design using DSCH Tool

Probability logic locking (PLL) is a technique used in digital circuit design to protect intellectual property by locking the circuit using a secret key. The idea is to modify the circuit such that it only works correctly when a specific input value, determined by the secret key, is provided.

PLL can be implemented using different logic gates such as NAND, NOR, AND, and extra key gates. Here is an explanation of how PLL can be implemented using these gates:

1. NAND Logic PLL: In this implementation, the circuit is modified by adding an extra NAND gate to each original gate. The extra NAND gate is connected to the original gate such that the output of the NAND gate becomes the input of the original gate. The secret key is used to control the inputs of the extra NAND gates. The extra NAND gates are connected in such a way that they cancel out the effect of the original gates when the wrong key is applied, and only the correct key combination can unlock the circuit.

2. NOR Logic PLL: In this implementation, the circuit is modified by adding an extra NOR gate to each original gate. The extra NOR gate is connected to the original gate such that the output of the NOR gate becomes the input of the original gate. The secret key is used to control the inputs of the extra NOR gates. The extra NOR gates are connected in such a way that they cancel out the effect of the original gates when the wrong key is applied, and only the correct key combination can unlock the circuit.
3. AND Logic PLL: In this implementation, the circuit is modified by adding an extra AND gate to each original gate. The extra AND gate is connected to the original gate such that the output of the AND gate becomes the input of the original gate. The secret key is used to control the inputs of the extra AND gates. The extra AND gates are connected in such a way that they cancel out the effect of the original gates when the wrong key is applied, and only the correct key combination can unlock the circuit.
4. Extra Key Gate PLL: In this implementation, an extra key gate is added to each input of the circuit. The extra key gate is a combination of different logic gates, such as NAND, NOR, AND, or XOR. The secret key is used to control the inputs of the extra key gates. The extra key gates are connected in such a way that they block the input signal when the wrong key is applied, and only the correct key combination can unlock the circuit.

In summary, PLL is a technique used to protect intellectual property in digital circuit design. It involves modifying the circuit using extra gates such as NAND, NOR, AND, and extra key gates, which can only be unlocked using a secret key.

V. SIMULATION RESULTS

A. UNLOCKED CIRCUIT

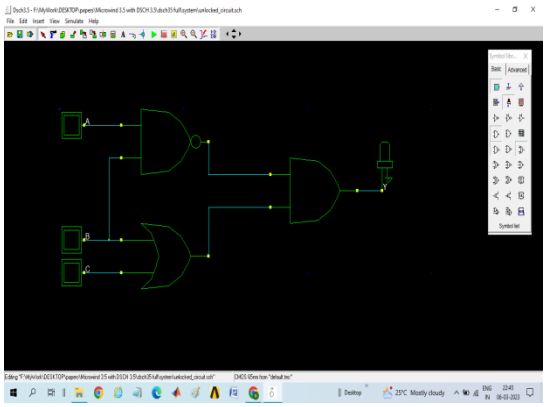


Fig.6: Design of Unlocked Circuit using DSCH Tool

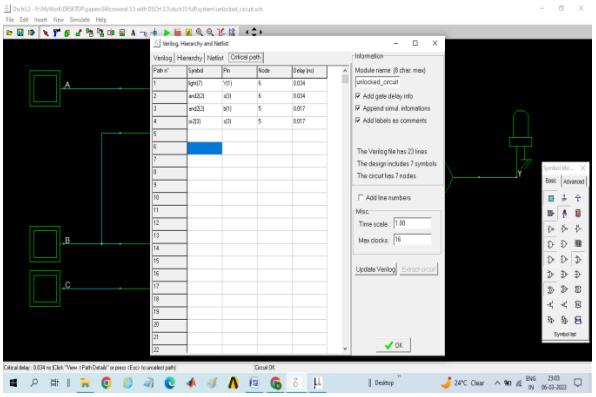


Fig.7: Showing Critical Path in DSCH Tool

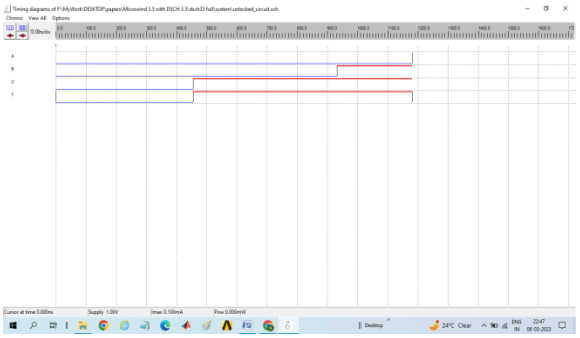


Fig.8: Timing Diagram on DSCH Tool

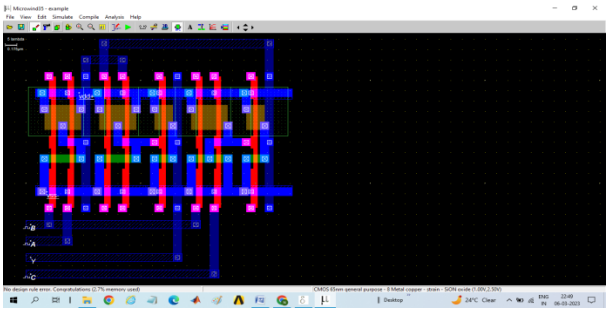


Fig.9: Generated Layout in Microwind Tool

Fig10: Output Waveform showing in Microwind Tool

B.LOCKING TECHNIQUE

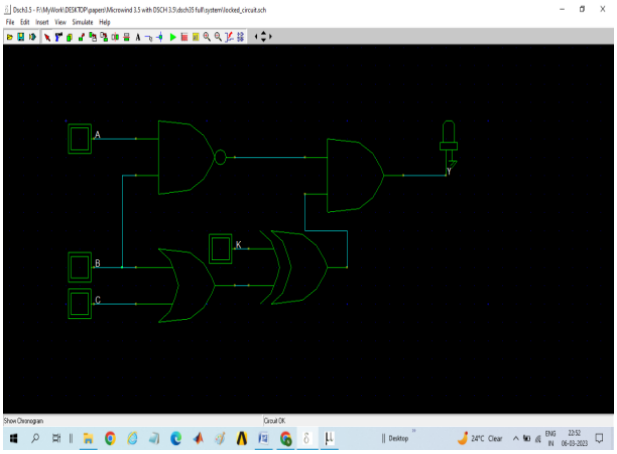


Fig.11: Design Circuit using DSCH tool

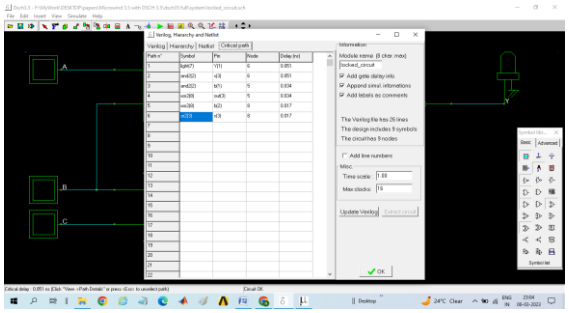


Fig.12: Showing Critical path in DSCH Tool

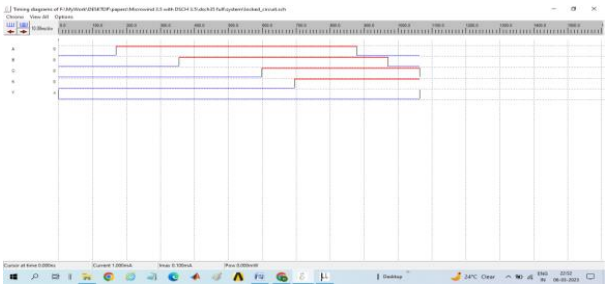


Fig.13: Timing Diagram shwoing in DSCH

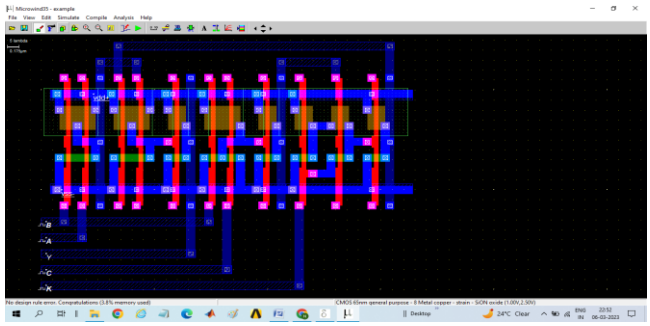


Fig.14: Layout

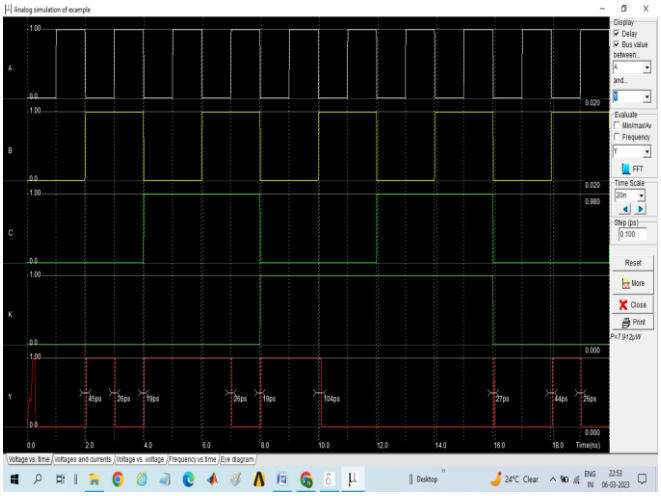


Fig.15: Layout Showing in Micro wind

VI. CONCLUSION AND FUTURE WORK

Probability-based logic-locking techniques using DSCH and Microwind tools can provide an effective solution for protecting intellectual property and preventing unauthorized access to circuit designs. By adding additional logic gates with probabilistic behavior, these techniques make it difficult for an attacker to reverse engineer the design and reproduce it without the proper key. Overall, the use of probability-based logic-locking techniques offers a promising approach to enhancing the security of integrated circuits, particularly for applications that require high levels of confidentiality and intellectual property protection. However, it is important to note that these techniques may also introduce additional design complexity and potentially impact the circuit's performance and power consumption. Therefore, careful consideration and evaluation are necessary to ensure that the benefits of logic-locking outweigh any potential drawbacks.

VII. FUTURE SCOPE

In the future, we intend to test the obfuscated benchmarks against known attacks and compare them to other logic locking techniques. We will implement logic locking attacks such as SAT attacks and sensitization attacks. Each attack will be executed against benchmarks obfuscated with ProbLock. We will then run the same attacks on locking schemes such as SLL, logic cone locking, and SARLock. We will evaluate how well each benchmark performs by measuring the overhead of the obfuscation technique, complexity of the technique, and execution time of the attack. After running each attack scheme, we can compare and evaluate the true strength of ProbLock compared to other published logic-locking techniques.

REFERENCES

1. Mellor, J.; Shelton, A.; Yue, M.; Tehranipoor, F. Attacks on logic locking obfuscation techniques. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 142–149.
2. Forte, D.; Bhunia, S. Hardware Protection through Obfuscation; Tehranipoor, M., Ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 1–349.
3. Lee, Y.; Touba, N.A. Improving logic obfuscation via logic cone analysis. In Proceedings of the 2015 16th Latin-American Test Symposium (LATS), Puerto Vallarta, Mexico, 25–27 March 2015; pp. 1–6.
4. Yasin, M.; Rajendran, J.J.V.; Sinanoglu, O.; Karri, R. On Improving the Security of Logic Locking. IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 2016, 1411–1424. [CrossRef]
5. Amir, S.; Shakya, B.; Xu, X.; Jin, Y.; Bhunia, S.; Tehranipoor, M.; Forte, D. Development and Evaluation of Hardware Obfuscation Benchmarks. J. Hardw. Syst. Secur. 2018, 142–161. [CrossRef]
6. Yasin, M.; Mazumdar, B.; Rajendran, J.J.V.; Jin, Y.; Sinanoglu, O. SARLock: SAT attack resistant logic locking. In Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 3–5 May 2016; pp. 236–241.
7. Tehranipoor, F.; Karimian, N.; Kermani, M.M.; Mahmoodi, H.; Sinanoglu, O. Deep rnn-oriented paradigm shift through bocanet: Broken obfuscated circuit attack. In Proceedings of the 2019 on Great Lakes Symposium on VLSI, Tysons Corner, VA, USA, 9–11 May 2019; pp. 335–338. [CrossRef]
8. Brglez, F.; Fujiwara, H. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan. In Proceedings of the International Symposium on Circuits and Systems, Kyoto, Japan, 5–7 June 1985; pp. 663–698.
9. Brglez, F.; Bryan, D.; Kozminski, K. Combinational Profiles of Sequential Benchmark Circuits. In Proceedings of the International Symposium on Circuits and Systems, Portland, OR, USA, 8–11 May 1989; pp. 1929–1934.
10. Yasin, M.; Mazumdar, B.; Rajendran, J.J.V.; Sinanoglu, O. TTLock: Tenacious and traceless logic locking. In Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2017), McLean, VA, USA, 1–5 May 2017; p. 166.
11. engupta, A.; Nabeel, M.; Yasin, M.; Sinanoglu, O. ATPG-based cost-effective, secure logic locking. In Proceedings of the 2018 IEEE 36th VLSI Test Symposium (VTS), San Francisco, CA, USA, 22–25 April 2018; pp. 1–6.
12. Yasin, M.; Sengupta, A.; Thari Nabeel, M.; Ashraf, M.; Rajendran, J.J.V.; Sinanoglu, O. Provably-Secure Logic Locking: From Theory To Practice. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and

Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1601–1618.
13. Yang, F.; Tang, M.; Sinanoglu, O. Stripped Functionality Logic Locking With Hamming Distance-Based Restore Unit (SFLL-hd) - 2013 Unlocked. IEEE Trans. Inf. Forensics Secur. 2019, 2778–2786. [CrossRef]
14. Yasin, M.; Zhao, C.; Rajendran, J.J.V. SFLL-HLS: Stripped-Functionality Logic Locking Meets High-Level Synthesis. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–4.
15. Saha, A.; Saha, S.; Bhattacharya, B.B.; Chowdhury, S.; Mukhopadhyay, D. Lopher: Sat-hardened logic embedding on block ciphers. In Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 20–24 July 2020; pp. 1–6.
16. Abdulrahman, A.; Bhunia, S. Scalable Attack-Resistant Obfuscation of Logic Circuits. arXiv 2020, arXiv:2010.15329.
17. Biere, A. Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016. Available online: fmv.jku.at/papers/Biere-SAT-Competition-2016-solvers.pdf (accessed on 5 October 2021).
18. Xie, Y.; Srivastava, A. Anti-SAT: Mitigating SAT Attack on Logic Locking. IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 2015, 38, 199–207. [CrossRef]
19. Miskov-Zivanov, N.; Marculescu, D. Modeling and Optimization for Soft-Error Reliability of Sequential Circuits. IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 2008, 27, 803–816. [CrossRef]
20. Waksman, A.; Suozzo, M.; Sethumadhavan, S. FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 697–708.
21. Design Compiler Graphical. Synopsys. 2018. Available online: https://www.synopsys.com/implementation-and-signoff/rtlsynthesis-test/design-compiler-graphical.html (accessed on 5 October 2021).
22. Subramanyan, P.; Ray, R.; Malik, S. Evaluating the security of logic encryption algorithms. In Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST2015), Washington, DC, USA, 5–7 May 2015; pp. 137–143.

Cite this article as :

P. Rajesh, Sompalli Charan Sai, Vellala Sai Sri Pranathi, Udatha Kavya Sree, Thippareddy Asuvardhan Reddy, A R Kushal, "Implementation of Logic-Locking Technique Based on Probability Using Back End Tool ", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 11 Issue 2, pp. 262-272, March-April 2024. Journal URL : https://ijsrst.com/IJSRST52411241