TimingCamouflage+ Decamouflaged

Priya Mittu
pmittu@umd.edu
Institute for Systems Research,
University of Maryland
College Park, Maryland, USA

Yuntao Liu
ytliu@umd.edu
Institute for Systems Research,
University of Maryland
College Park, Maryland, USA

Ankur Srivastava
ankurs@umd.edu
Institute for Systems Research,
University of Maryland
College Park, Maryland, USA

ABSTRACT

In today's world, sending a chip design to a third party foundry for fabrication poses a serious threat to one's intellectual property. To keep designs safe from adversaries, design obfuscation techniques have been developed to protect the IP details of the design. This paper explains how the previously considered secure algorithm, TimingCamouflage+, can be thwarted and the original circuit can be recovered [15]. By removing wave-pipelining false paths, the TimingCamouflage+ algorithm is reduced to the insecure TimingCamouflage algorithm [16]. Since the TimingCamouflage algorithm is vulnerable to the TimingSAT attack, this reduction proves that TimingCamouflage+ is also vulnerable to TimingSAT and not a secure camouflaging technique [7]. This paper describes how wave-pipelining paths can be removed, and this method of handling false paths is tested on various benchmarks and shown to be both functionally correct and feasible in complexity.

CCS CONCEPTS

• **Security and privacy → Hardware reverse engineering**.

KEYWORDS

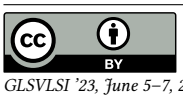
Camouflaging, Wave-pipelining, False paths, TimingSAT

ACM Reference Format:
Priya Mittu, Yuntao Liu, and Ankur Srivastava. 2023. TimingCamouflage+ Decamouflaged. In *Proceedings of the Great Lakes Symposium on VLSI 2023 (GLSVLSI '23), June 5–7, 2023, Knoxville, TN, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3583781.3590238

1 INTRODUCTION

Security has become a major concern in today's production of integrated circuits (ICs). The vast majority of IC designers do not own or have access to their own foundries, so they have to send their designs to untrusted third-party foundries for fabrication, which introduces security risks. A third-party foundry may have adversaries who attempt to steal design information, overproduce chips to sell for a profit on the side, or introduce malicious hardware that would allow one to read secret data from a chip obtained legitimately. In order to prevent an adversary's malicious intentions and protect one's intellectual property (IP), a designer will obfuscate their design before sending it to an untrusted third party for fabrication.

This paper will provide an overview of previously proposed obfuscation algorithms and the attacks that have been developed to overcome them, as well as a detailed examination of TimingCamouflage+, a camouflaging algorithm that is previously considered secure . The paper will describe an attack that shows TimingCamouflage+ is not a secure camouflaging algorithm, and present experimental results to support this conclusion [15].



GLSVLSI '23, June 5–7, 2023, Knoxville, TN, USA

ACM ISBN 979-8-4007-0125-2/23/06.
https://doi.org/10.1145/3583781.3590238

2 BACKGROUND

2.1 Obfuscation Techniques

Most camouflaging techniques use various circuitry characteristics, such as threshold voltage or contacts between metal layers, to obscure gate functionality [2–4, 8]. One proposed locking scheme inserts logic barriers into a circuit such that every path from an input to an output passes through a logic barrier [1]. There are also many metering schemes that have been proposed [6, 10]. Metering schemes are locking techniques that insert key-gates (gates that take a key bit as input) in such a way that different chips will require different keys.

These logic obfuscation techniques above are vulnerable to SAT (boolean satisfiability) attacks [11]. The SAT attack identifies special input patterns that allow groups of key values to be ruled out. Input patterns are selected iteratively until a correct key value has been found. There are also SAT attack formulations specifically for camouflaging as opposed to locking [5].

Researchers have proposed SAT attack resilient camouflaging techniques, including AND-Tree based camouflaging and CamoPerturb [8, 13]. These techniques force the SAT attack to take exponential time to finish, similar to SAT resilient logic locking approaches such as Anti-SAT and Stripped Functionality Logic Locking (SFLL) [12, 14]. However, as long as SAT attacks can be formulated, it has been proved that camouflaging or logic locking approaches must have very low error impact if they want to achieve SAT resiliency [9].

Timing based camouflaging is superior to the above mentioned logic based camouflaging in that it obfuscates the timing, rather than the logic, of the circuit. Therefore, the logic based SAT attack cannot be formulated against timing based camouflaging. This paper focuses on the evolution of timing based camouflaging techniques and formulates an attack methodology against TimingCamouflage+, the state of the art timing based camouflaging technique [15].

2.2 TimingCamouflage

The TimingCamouflage algorithm creates wave-pipelining true paths in a circuit by removing specific flip flops [16]. These paths are created in such a way that they are topologically indistinguishable from conventional true paths. Details on how these paths are

created are the same as those described in Section 3. By not knowing whether a path is conventional or wave-pipelining, the correct functionality of the circuit is unknown by inspecting the netlist. A SAT based attack could be used to identify which paths are wave-pipelining (*e.g.* wave-pipelining paths were represented by a key bit of 1 and conventional paths were represented by a key bit of 0), but would not be able to identify where the flip flop was removed from. This is important as different locations of the flip flop can produce different functionalities. In this way, TimingCamouflage is secure from standard SAT based attacks.

2.3 TiminigSAT Attack

To identify the flip flop removal locations from wave-pipelining true paths, a new SAT-based attack that focuses on timing, TimingSAT, was proposed [7]. TimingSAT relies on inserting transformation units (TUs), made of a flip flop and a multiplexer, into the camouflaged circuit to convert wave-pipelining paths to conventional paths. The TUs are inserted into each potential flip flop location and introduce key bits into the circuit. Depending on the value of the TU's key, the flip flop in the TU will either be functional or not. At this point, a regular SAT based attack can be applied to determine a correct key. The TUs that have a key value of 1 mark the locations of where flip flops should be reinserted to regain original functionality. In this way, TimingSAT renders the TimingCamouflage algorithm unsecure. In an attempt to make the TimingCamouflage algorithm secure, wave-pipelining false paths were introduced, and the TimingCamouflage+ algorithm was proposed and claimed to be secure against any known attack [15].

3 TIMINGCAMOUFLAGE+: AN IMPROVEMENT ON TIMINGCAMOUFLAGE

3.1 Types of Combinational Paths

The TimingCamouflage+ algorithm is built on creating different kinds of paths, specifically various combinations of true/false paths and conventional/wave-pipelining paths. Traditionally, a circuit is made up of only conventional paths, meaning all paths have a delay less than one clock cycle. A wave-pipelining path has a delay greater than one clock cycle, and hence multiple data packets propagate down the path at the same time. In TimingCamouflage+, wave-pipelining paths will have a delay time greater than one clock cycle and less than or equal to two clock cycles. Figure 1a shows an example of a wave-pipelining path. If at time 0 the signal B changes from 0 to 1, the signal will not propagate to location C in time to be reflected in the output D at time T, where T represents the time of one clock cycle (this will be the case throughout this paper). The propagation of the signal can be seen in Figure 1b. The transition of signal B from 0 to 1 at time 0 is reflected in the output D at time 2T. If signal B changes at time T, then there will be two signal changes being propagated in the path between time T and 2T in a pipelined fashion.

In the scope of this paper along with the TimingCamouflage+ paper, a false path is defined as a combinational path that cannot be sensitized because of controlling signals from other paths. Figure 2a shows a false path highlighted in blue. Notice that signal A does not affect the result at signal C. When signal B is 1, it controls the OR gate causing C to get a value of 1. When signal B is 0, it controls

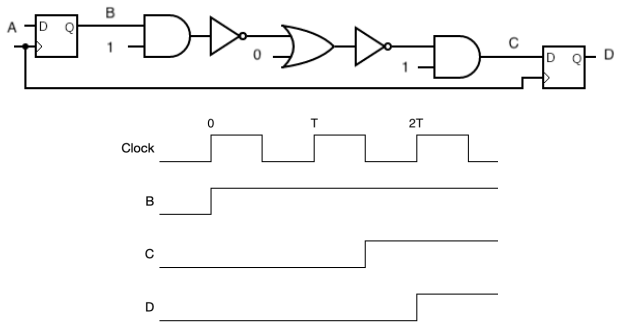


Figure 1: Wave-pipelining example.

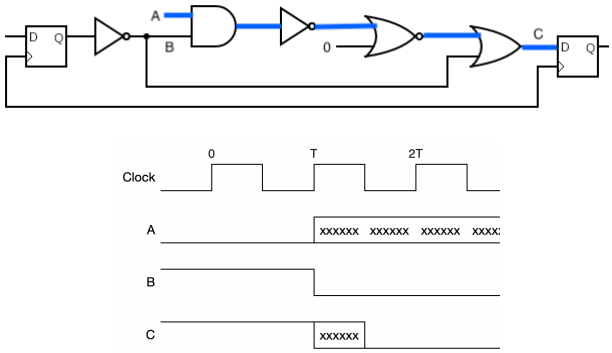


Figure 2: False path example.

the AND gate which causes C to get a value of 0. Regardless of what value it is, signal B controls the result at C. In this case, B is the controlling signal. A path that is not false is called true.

Conventional true and false paths follow directly from the definitions above. However, wave-pipelining false paths are defined as wave-pipelining paths that are false when viewed as conventional paths. If figure 2a was a wave-pipelining false path, signal A could affect the result at C. Consider the signal propagation in figure 2b. When signal B is 1 in the first clock cycle, C gets a value of 1. At time T, B becomes 0 and the value of C, once stabilized, depends on the value of A from the previous clock cycle. If the value of A was 0, C will be 0, and if the value of A was 1, C will be 1. In this way, signal A is able to affect the resulting signal at C. The definition of a wave-pipelining true path is synonymous with the definition of a wave-pipelining false path.

3.2 Algorithm

To obfuscate a circuit using TimingCamouflage+, wave-pipelining paths must be created by removing specific flip flops. Note that it is possible to remove a flip flop and have the resulting path still be conventional. However, these flip flops would not be considered eligible to be removed. Additionally, removing a single flip flop can affect many paths as it can have many fanin and fanout paths.

Because many circuits may not have candidate flip flops only belonging to one path, instead of simply removing the candidate flip flop, the circuit surrounding the flip flop is duplicated so that some paths may use the flip flop and others may have it removed. Duplicating the surrounding circuit also prevents creating too many wave-pipelining paths all in one place.

Creating wave-pipelining paths is not enough on its own to obfuscate a circuit. An attacker could simply perform a static timing analysis to find the wave-pipelining paths and regain the original circuit. However, the TimingCamouflage+ algorithm uses the notion of a "grey region" to help prevent this attack. Taking into account an error value δ, the grey region is defined as the values between T-δ and T+δ, where T is the period of one clock cycle. After performing static timing analysis, the paths that fall within the grey region cannot be confidently categorized as either conventional or wave-pipelining. Then, the attacker must use input vectors to test whether a certain path truly is wave-pipelining by comparing results to an oracle. When removing flip flops, the goal is to have the newly created wave-pipelining paths, in addition to some conventional paths, fall in the grey region. This is to prevent an attacker from assuming that all grey region paths are wave-pipelining.

Finally, in order for the camouflaging to be secure, both true and false wave-pipelining paths need to be created. As noted above, true wave-pipelining paths are not fully secure as they can be sensitized by certain input vectors. However, false wave-pipelining paths cannot be sensitized by an input vector. Notice that when using input vectors to try to sensitize a path, the paths are viewed as conventional because scan chain access is used for sensitization. When scan chain access is being used, only one input vector can be used (i.e. an input vector is given, and after a specified amount of time, the content of the scan chain is read). While a false wave-pipelining path may be sensitizable in reality, it is not sensitizable by an input vector using scan chain access. To be sensitized, a false wave-pipelining path requires 2 consecutive input vectors which is not supported by scan chain access. The authors of the TimgingCamouflage+ paper claim that in order for these paths to be found, major changes would need to be made to the scan chain allowing for multiple clock periods to be observed [15]. By including the 4 types of paths, the TimingCamouflage+ paper claims its camouflaging algorithm is secure.

4 HANDLING WAVE-PIPELINING FALSE PATHS

4.1 Overview

As acknowledged above, there is already a known method to discover true wave-pipelining paths, so this paper will only discuss the false wave-pipelining path case. Once this case is handled, TimingCamouflage+ will no longer be considered a secure camouflaging scheme.

4.2 Controlling Signals

False paths are dependent on the types of gates they are made up of and the orientation of the gates. A controlling signal of a gate is an input that determines the output of that gate regardless of what the other input values are. For example, the controlling signal of an AND gate is 0, because whether the other inputs are 0s or 1s, the output will always be 0. Likewise, the controlling signal of an OR gate is 1. There isn't the notion of a controlling signal in a NOT gate, because there is only 1 input signal.

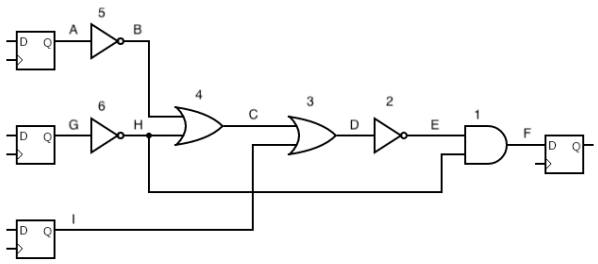
Similar to a gate's controlling signal, a controlling signal of a path is a signal that always determines the output of that path regardless of what the other input values are. In figure 2a, B is the controlling signal as it always controls the output. Notice that the controlling signal in this case is connected to two gates with opposite controlling values. This allows signal B to be the path's controlling signal as it will always control one of the two gates. This setup of two gates with opposite controlling values on the same path with the same input signal is needed for the path to have a controlling signal (namely the shared signal).

Controlling signals cause adjacent signals to become expendable and unsensitizable, as the adjacent signals have no impact on the output. This leads them to become false paths. In figure 2a, signal A is adjacent to the controlling signal B and produces the false path highlighted in blue as it can never be sensitized.

The first step in handling false wave-pipelining paths is to identify controlling signals. This is done by doing a reverse topological search on all the paths in the given camouflaged circuit. Starting at either a flip flop input or an output of the circuit, paths are traced backwards until a flip flop output or circuit input is reached. While paths are being traced, the types of gates are recorded along with their input signals. Once Two gates with opposing controlling values that share an input are found on the same path, that shared signal is returned as a controlling signal. Figure 3 shows an example circuit along with its tracing. Tracing begins at the last flip flop's input and works towards each of the previous flip flops' output. At each gate encountered, the gate along with both its input signals are added to the set of things seen. Note that the set of things seen is path specific, so it is duplicated at a fork. For example, at the AND gate, the set of things seen is copied for both input signals, leading to locations two and six, and then modified accordingly. At any point in the circuit, the set of things seen only includes things topologically in front of the current point. Also note that the NOT gate at location six is visited twice. This is because its output, signal H, drives two separate gate inputs. In other words, it is on two paths and must be handled twice. Once location four is reached, two gates with opposing controlling values that share an input are found. This can be seen highlighted in green in figure 3b, and at this point, signal H would be returned as a controlling signal.

4.3 Identifying Wave-Pipelining False Paths

Once controlling signals have been identified, the corresponding false paths are also found. The next step is to classify these false paths as either conventional or wave-pipelining. This can be challenging, because conventional false paths can appear longer than a single clock cycle. Consider the false path in Figure 4. If the path highlighted in blue is wave-pipelining, but the path highlighted in green is conventional, then the blue path is still considered a conventional false path. This is because the output at C stabilizes within one clock cycle, and any change in signal A will not ever be reflected in the output at C. In this case, the timing (and functionality) of the path highlighted in blue is irrelevant.



	Location						
	1	6	2	3	4	5	6
Seen	AND, E	AND, E	AND, E	AND, E	AND, E	AND, E	AND, E
	AND, H	AND, H	AND, H	AND, H	AND, H	AND, H	AND, H
		NOT, G	NOT, D	NOT, D	NOT, D	NOT, D	NOT, D
				OR, C	OR, C	OR, C	OR, C
				OR, I	OR, I	OR, I	OR, I
					OR, B	OR, B	OR, B
					OR, H	OR, H	OR, H
						NOT, A	NOT, G

Figure 3: Control signal tracing example.

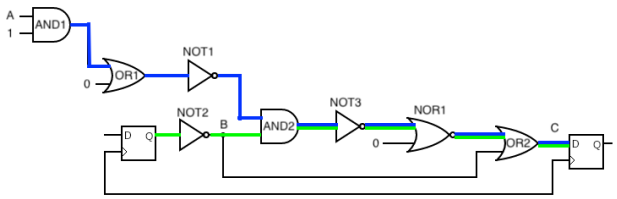


Figure 4: An example of a conventional false path that has more than 1 clock cycle of delay

However, if the path highlighted in green is wave-pipelining, then the path in blue becomes a wave-pipelining false path. In order for a false path to be wave-pipelining, the path of the controlling signal must be wave-pipelining. This provides the opportunity for the controlling signal to become non-controlling in certain cases. This is similar to the example in figure 2.

To identify these wave-pipelining false paths, all paths containing controlling signals need to be analyzed for timing. This can be done by setting all side inputs to non-controlling values and performing a static timing analysis to see how long it takes the controlling signal to propagate to the output. If one of the controlling paths is multi-cycled, then the corresponding wave-pipelining false path is found.

4.4 Reducing Wave-Pipelining False Paths to Wave-Pipelining True Paths

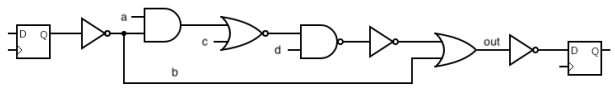
Once wave-pipelining false paths have been found, flip flops should be reinserted to uncamouflage the circuit. However, flip flop reinsertion creates another challenge as the exact location of flip flop insertion can be difficult to identify, but necessary for correct functionality as different flip flop locations can produce different functionalities.

However, this challenge of finding flip flop reinsertion locations is not unique to false wave-pipelining paths. In fact, the same difficulty is also present in true wave-pipelining paths which have already been acknowledged as not secure. Therefore, to handle false wave-pipelining paths, they need only be converted to true wave-pipelining paths. To do this, the controlling path of the wave-pipelining false path will be duplicated and replaced to explicitly show the hidden wave-pipelining true paths. Consider the false wave-pipelining path in figure 5a. This circuit's functionality can be broken down into four cases depending on two recent values of the controlling signal. These cases are categorized in a truth table in figure 5b. The first two cases, when b[t=0]=1, have the same result, so they can be generalized to one case — leaving three cases in total. The controlling path of the original wave-pipelining false path can be replaced with a 3x1 multiplexer with the three out cases as inputs and two recent values of the controlling signal as the select lines. The two values of the controlling signal used are taken at time t=0 and t=-x, where x is the time it takes the controlling signal to arrive at the last controlling gate in the original camouflaged circuit. This new circuit can be seen in figure 5c. Here the original circuit has been converted into an equivalent circuit with new true paths (one of which is highlighted in blue). The blue path is guaranteed to be a true wave-pipelining path, because it was the original false wave-pipelining path. The green path may or may not be wave-pipelining depending on the location of the removed flip flop in the original circuit, but this information is not needed. Since they yield the same truth table, this conversion is guaranteed to have the same functionality as the original circuit. This process can be generalized to work for all wave-pipelining false paths. At this point, the wave-pipelining true paths can be converted back to conventional paths using the TimingSat algorithm as described in Section 2 [7].

5 EXPERIMENTAL RESULTS

5.1 Overview

In order to evaluate the algorithm presented above, it was first implemented, then functionality was confirmed, and finally runtime was measured. For camouflaged circuits, the TimingCamouflage+ algorithm was simulated and ISCAS89 benchmarks were made to be consistent with that of the output of TimingCamouflage+. Next, controlling signals, along with their gates, were identified by tracing. Using the controlling signals and gates, controlling paths were also identified by tracing. A timing analysis was done on these controlling paths in order to classify them as conventional or wave-pipelining. Grey paths are treated as wave-pipelining, because if they are conventional, TimingSAT will find no flip flop locations resulting in no issues. On the other hand, if a grey path was treated as a conventional path but was wave-pipelining, the path would not be uncamouflaged. Once controlling paths have been identified, they can be replaced with their equivalent true paths rendering the circuit unsecure.



b[t=0]	b[t=-1]	out
1	1	1
1	0	1
0	1	!(!(!(a || c) && d))
0	0	!(!(!c && d))

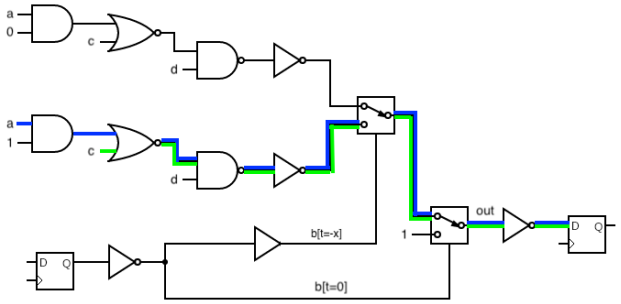


Figure 5: Wave-pipelining false path removal example.

5.2 Functionality

Functionality checking is provided here for the smallest benchmark, s27. As seen in figure 6a, signal DFF_0.Q was identified as a controlling signal of a wave-pipelining path, and the controlling path associated with this controlling signal was found: _17_, _07_, _18_. At this point, the last gate of the controlling path, _18_, was replaced with a 3x1 multiplexer and the rest of the gates were duplicated. Since the controlling signal took 0.7T to propagate to the last controlling gate in the original circuit, the select lines to the 3x1 mux are the current controlling signal and the controlling signal delayed by 0.7T. The resulting circuit is shown in figure 6b. The waveforms for the camouflaged and uncamoflagued circuits are depicted in figure 7. As shown, the two inputs to DFF_1._2_, DFF_1.D and DFF_1.D2, are consistent with each other. As we have examined all four possibilities of control signal values within two cycles, the circuits before and after the conversion of wave-pipelining false paths to wave-pipelining true paths are equivalent, and the conversion is valid.

5.3 Runtime

This process was performed on five benchmarks of varying size. Details of these benchmarks can be found in figure 8. This table shows that the algorithm's execution time increases with the number of false paths and not with the total size of the circuit as can be seen by benchmarks s1196 and s1423. Additionally, the time needed to handle false wave-pipelining paths is relatively small which means this is a feasible attack algorithm.

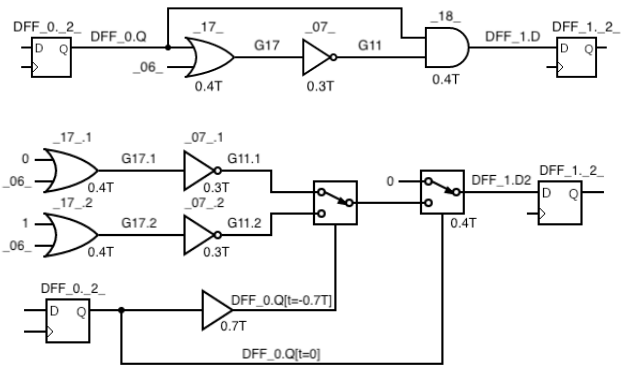


Figure 6: Removal of wave-pipelinig false paths from benchmark s27.

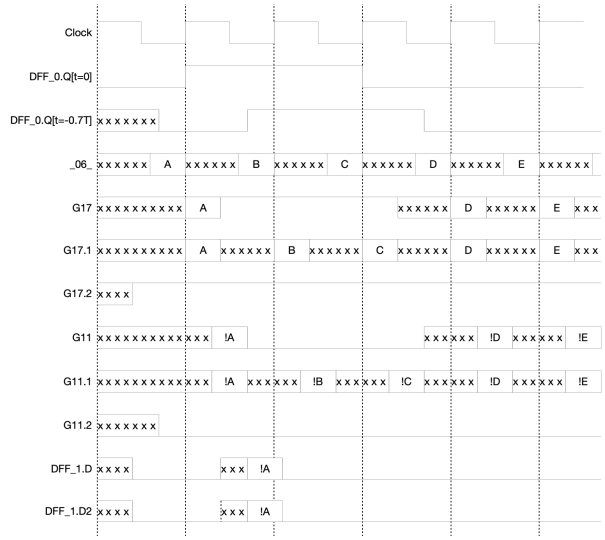


Figure 7: Functional verification of benchmark s27; waveforms of 6(a) and (b).

Benchmark	# Flip Flops	# Gates	# Conventional False Paths	# Wave-Pipelining False Paths	Execution Time (ms)
s27	3	10	1	1	0.05536
s382	21	158	1	2	0.22442
s526	21	193	3	2	0.26674
s1196	179	2779	7	5	0.71388
s1423	74	657	27	17	12.43602

Figure 8: The results of handling wave-pipelining false paths in five ISCAS89 benchmarks.

6 CONCLUSION

In this paper, the security of the TimingCamouflage+ algorithm is examined. While it is argued to be secure because of its unidentifiable wave-pipelining false paths, a method is proposed that allows these paths to be converted back to their original conventional form. In order to do this, controlling signals are first found, controlling paths are then identified, and finally controlling paths are duplicated in a way that converts wave-pipelining false paths to wave-pipelining true paths. At this point, the TimingSAT algorithm can be used to convert wave-pipelining true paths back to conventional paths. By performing these path conversions, this paper has demonstrated that the TimingCamouflage+ algorithm is not a secure camouflaging technique as it is vulnerable to the TimingSAT algorithm after manipulation.

ACKNOWLEDGMENTS

This work was supported by the Air Force Research Laboratory under the Locked Electronics for Assured Design (LEAD) project FA8650-18-S-1201.

REFERENCES

[1] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE design & Test of computers* 27, 1 (2010), 66–75.

[2] Ronald P Cocchi, James P Baukus, Lap Wai Chow, and Bryan J Wang. 2014. Circuit camouflage integration for hardware IP protection. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–5.

[3] Maria I Mera Collantes, Mohamed El Massad, and Siddharth Garg. 2016. Threshold-dependent camouflaged cells to secure circuits against reverse engineering attacks. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 443–448.

[4] Giovanni Di Crescenzo, Jeyavijayan Rajendran, Ramesh Karri, and Nasir Memon. 2017. Boolean circuit camouflage: Cryptographic models, limitations, provable results and a random oracle realization. In *Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security*. 7–16.

[5] Mohamed El Massad, Siddharth Garg, and Mahesh V Tripunitara. 2019. The SAT attack on IC camouflaging: Impact and potential countermeasures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 8 (2019), 1577–1590.

[6] Farinaz Koushanfar. 2011. Provably secure active IC metering techniques for piracy avoidance and digital rights management. *IEEE Transactions on Information Forensics and Security* 7, 1 (2011), 51–63.

[7] Meng Li, Kaveh Shamsi, Yier Jin, and David Z Pan. 2018. TimingSAT: Decamouflaging timing-based logic obfuscation. In *2018 IEEE International Test Conference (ITC)*. IEEE, 1–10.

[8] Meng Li, Kaveh Shamsi, Travis Meade, Zheng Zhao, Bei Yu, Yier Jin, and David Z Pan. 2017. Provably secure camouflaging strategy for IC protection. *IEEE transactions on computer-aided design of integrated circuits and systems* 38, 8 (2017), 1399–1412.

[9] Yuntao Liu, Michael Zuzak, Yang Xie, Abhishek Chakraborty, and Ankur Srivastava. 2020. Strong anti-SAT: Secure and effective logic locking. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 199–205.

[10] Jarrod A Roy, Farinaz Koushanfar, and Igor L Markov. 2010. Ending piracy of integrated circuits. *Computer* 43, 10 (2010), 30–38.

[11] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 137–143.

[12] Yang Xie and Ankur Srivastava. 2018. Anti-sat: Mitigating sat attack on logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 2 (2018), 199–207.

[13] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. 2016. CamoPerturb: Secure IC camouflaging for minterm protection. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[14] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan Rajendran, and Ozgur Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1601–1618.

[15] Grace Li Zhang, Bing Li, Meng Li, Bei Yu, David Z Pan, Michaela Brunner, Georg Sigl, and Ulf Schlichtmann. 2020. TimingCamouflage+: Netlist security enhancement with unconventional timing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4482–4495.

[16] Grace Li Zhang, Bing Li, Bei Yu, David Z Pan, and Ulf Schlichtmann. 2018. TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 91–96.